cucai 2023

Proceedings.

Canadian Undergraduate Conference on Artificial Intelligence

March 4–5 | Kingston ON

Foreword

The Canadian Undergraduate Conference on Artificial Intelligence (CUCAI) was hosted on March 4–5, 2023 in Kingston, ON, making its 1st in-person return since 2020. Since its inception, CUCAI has become the most prestigious undergraduate conference in Canada, brining together 335 bright minds this year. Highlights from this weekend include keynotes by Ivan Zhang, Raymond Lo, Wemba Opota, Allison Cohen, Iman Isla Bashir, & Nicole Lytle, workshops from RBC, Deloitte, and Accenture, as well as our flagship AI research showcase. During the showcase, students presented their findings in booths to conference attendees, industry sponsors, and top-level recruiters. Participating organizations included QMIND from Queen's University, Western AI from Western University, Wat.ai from the University of Waterloo, UVicAI from the University of Victoria, McGill AI Society from McGill University, UofT AI from the University of Toronto, & AIYU from York University.

We look forward to seeing more universities partake in CUCAI 2024. To find more about our mission, please visit: ***www.cucai.ca***

Awards

Of the 25 submissions, a shortlist of the top 10 papers were selected and forwarded to our selections team comprising of academics in the field of AI. Awards were evaluated on the basis of quality of work, novelty of the approach & goal, and overall quality of the paper.

Thank you to Ning Lu, Yuan Tian for your time and expertise in contributing to Canadian Undergraduate Conference on Artificial Intelligence.

Top Submissions

Wearable Brain-Computer Interface Paired with Large Language Model for Fast Speech Communication

Creating a Corn Yield Estimator: A Deep Learning Approach via UAV Imagery

Improving Public Transit Systems through Clustering and Polynomial Regression

Honourable Mentions

Quantum Convolutional Neural Networks

Monte Carlo Tree Search and Reinforcement Learning for a Four Player, Simultaneous Move Game

AGORA: a Language Model for Safe Speech-to-Text Conversion

Table of Contents

Table of Contents

Sponsor Acknowledgement

We are forever grateful of all the support from our sponsors and industry partners who partook in executing the Canadian Undergraduate Conference on Artificial Intelligence 2023. Without this level of support, we wouldn't have been able to bring together 335 of Canada's brightest undergraduate talent to experience a weekend of keynotes, workshops, networking sessions, project showcases, and interactive Q&A sessions. The delegate experience at CUCAI 2023 was unforgettable; from the company booths, to the sponsored lunches, and the mentoring/guidance from industry leaders.

We would like to extend a special thank you to the Faculty of Engineering & Applied Science at Queen's University, who made a huge portion of this event possible and has actively supported both QMIND and CUCAI's core mission from inception. We were both proud and excited at the opportunity to bring CUCAI back to the Kingston area for the first time going virtual in 2020. We would also like to thank the National Research Council of Canada for supporting this year's initiative.

With the continuous support from global corporations, we hope to continue growing CUCAI's prestige and reputation in the AI space. We thank all sponsors for contributing to our mission.

co:here

aws

RBC

BOREALIS AI
RBC Institute for Research

accenture

Deloitte.
OmniaAI

Vennify AI

Dunin-Deshpande
Queen's INNOVATION CENTRE

VECTOR INSTITUTE

Canada
National Research Council Canada

Queen's UNIVERSITY | FACULTY OF ENGINEERING AND APPLIED SCIENCE

Organizers

The Canadian Undergraduate Conference on Artificial Intelligence 2023 was organized by a team of exceptional undergraduates across various Canadian universities & institutions.

Leadership

Cooper Lloyd *Co-Chair*

Gonzalo Soto *Co-Chair*

Chris Gauthier *Co-Chair*

Spencer Hill *Advisor*

Olivia Xu *Advisor*

Marketing

Lejla Sain *Designer*

Alwin Raymundo *Designer*

Claire Cunningham *Social Media Coordinator*

Aren Jo *Photography & Video*

Jason Wei *Video Editor*

Partnerships

Marcelo Chaman *Coordiantor*

Leo Mckee–Reid *Coordinator*

Jarett Dewbury *Speakers*

Logistics

Tung Pham *Coordinator*

Adi Groumoutis *Delegates*

Rabab Azeem *Volunteer*

Harley Latsky *Volunteer*

Wearable Brain-Computer Interface Paired with Large Language Model for Fast Speech Communication

Christopher Samra
University of Waterloo
csamra@uwaterloo.ca

Livia Murray
University of Waterloo
l8murray@uwaterloo.ca

Dhrumil Patel
University of Waterloo
dhrumil.patel@uwaterloo.ca

***Abstract*—Current non-invasive wearable brain-computer interfaces for communication are intended for people with neurological conditions who cannot otherwise communicate. One of the largest challenges with these existing solutions is that their information transfer rates are much lower than verbal speech. This limits the quality of life of those who use these interfaces as it prevents them from speaking as quickly as they would like to, and in many cases conversations and social interactions that are time sensitive unfairly exclude these users from participation. To minimize this problem, our team built an interface that pairs existing techniques in literature (Electroencephalography, Steady State Visually Evoked Potentials), with new advancements in Machine Learning (Large Language Models) to provide intelligent phrase suggestions based on conversational context. Preliminary results show our implementation has enabled people using our interface to communicate over two times faster than anything we could find in existing literature.**

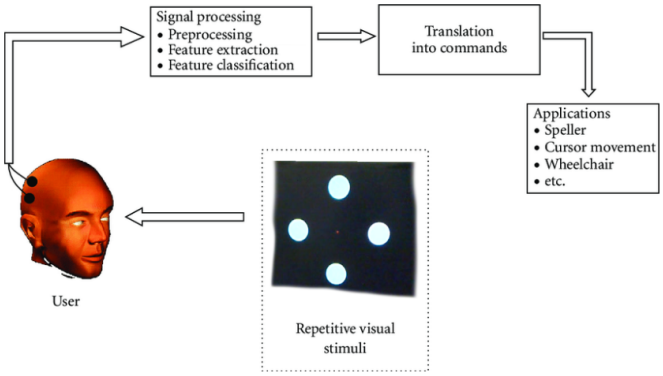

Fig. 1. SSVEP Interface Diagram

I. Introduction

A. Motivation

Brain computer interface (BCI) technology can provide a new channel of communication to humans, especially those with severe disabilities, such as stroke, spinal cord injury and amyotrophic lateral sclerosis (ALS) [1].

Many wearable BCI systems use electroencephalography (EEG) to measure brain activity. EEG measures brain activity non-invasively by recording electrical activity at the scalp that arises from large activations of brain cells [2]. A common paradigm used in EEG BCI literature is known as Steady State Visually-Evoked Potentials (SSVEP). SSVEP is a type of brain activity that occurs in response to a continuous visual stimulus flickering at a specific frequency [3]. SSVEPs are used as input signals to allow users to control elements of a user interface on a screen [3]. The basic principle behind SSVEP is that when a person focuses their attention on a visual stimulus that flickers at a specific frequency, their visual cortex produces a corresponding electrical signal at the same frequency [3].

One of the common applications for SSVEP interfaces include keyboards, more commonly known as SSVEP spellers [3].

B. Related Works

Traditionally, SSVEP speller designs consist of targets of various alphanumeric characters that are each assigned different frequencies of stimuli. These characters are selected when the user focuses on the stimuli associated with them [3]. However, while this may be acceptable for texting, typing character by character may not be ideal when BCI users want to speak to others verbally.

One implementation of an SSVEP speller attempted to use autocomplete to rectify this problem, but in their testing the researchers did not see information transfer rate (ITR) scores higher than 78 bits/min [4], this translates to around 5 words per minute [5].

The best case example of an SSVEP interface using characters to type that we could find yielded an average ITR of 151 bits/min [6], this translates roughly to 10 words per minute [5].

These ITRs are too slow to catch up with verbal communication rates, especially considering the average conversational verbal speech rate is between 120-150 words per minute [7].

C. Problem Definition

Almost all existing SSVEP speller applications that our team reviewed have proven to be not suitable for verbal communication due to their design restricting ITR.

SSVEP spellers have design limitations due to physical screen space, and the range of most usable stimuli frequencies (8-18hz) [8]. Because of these restrictions, only a limited number of interactable keys (buttons with stimuli) can be present at a given time, meaning character by character typing can be time consuming. This low ITR problem has a huge impact when people with neurological conditions attempt to communicate in time-sensitive social situations, and they can often feel left behind. In the next section we discuss the methodology we employ to build out our own SSVEP interface and how we use a Large Language Model (LLM) to improve ITR.

II. Methodology

A. BCI Design

For our EEG device, our team used a gtec Unicorn [9], we designed custom mounts for the electrodes to fit inside a modified OpenBCI UltraCortex Mk4 headframe [10], these design decisions allowed us to take advantage of the gtec's better signal clarity in a head-frame that could be fitted to several different head sizes.

The eight EEG channels provided by the device were placed on the PO4, O2, OZ, POZ, PO3, O1, PO8, PO7 EEG Electrode locations [11], chosen for proximity to the occipital region (the region responsible for processing visual stimulus).

Eight stimuli were chosen with the following frequencies 11.75, 12.75, 10.25, 14.75, 11.25, 8.25, 10.75 and 13.25 hz. These stimuli were chosen arbitrarily within the 8 to 15Hz range as this range yielded the best results in our testing.

Fig. 2. Subject Wearing EEG Headset

B. Data Collection and Validation

SSVEP data was collected and visualized offline to validate signal quality. This was done by applying a bandpass filter from 6 to 18 hz and visualizing the signals in two ways.

The first being a plot of the Fast Fourier Transform (FFT) of the EEG signal averaged across trials from a particular stimuli frequency and EEG channel, the second validation method involved taking the first Principal Component Analysis (PCA) component from one trial of a stimuli frequency across all channels. The response to the stimuli in the SSVEP data collection method was validated as being correct with a comparison between these two approaches.

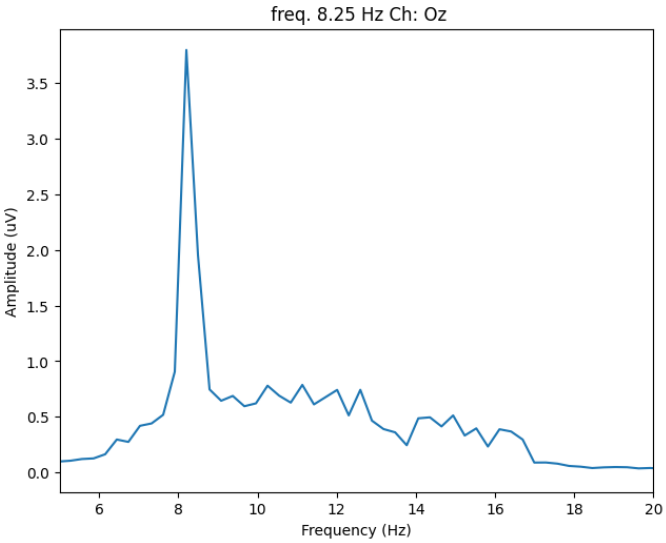

Fig. 3. FFT Averaged Across Trials

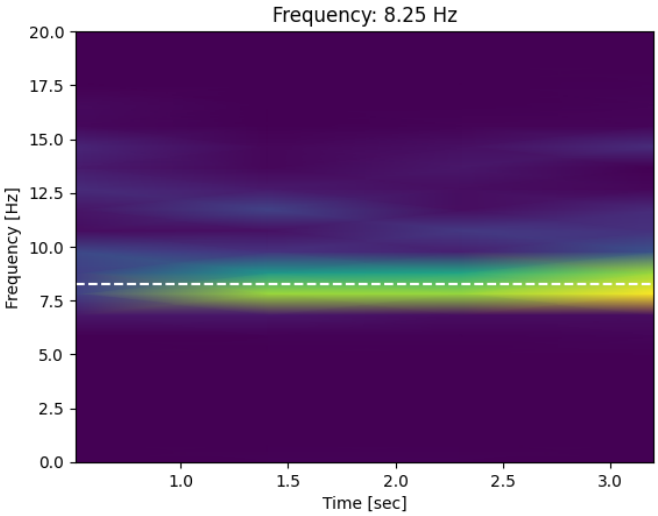

Fig. 4. PCA Across EEG Channels

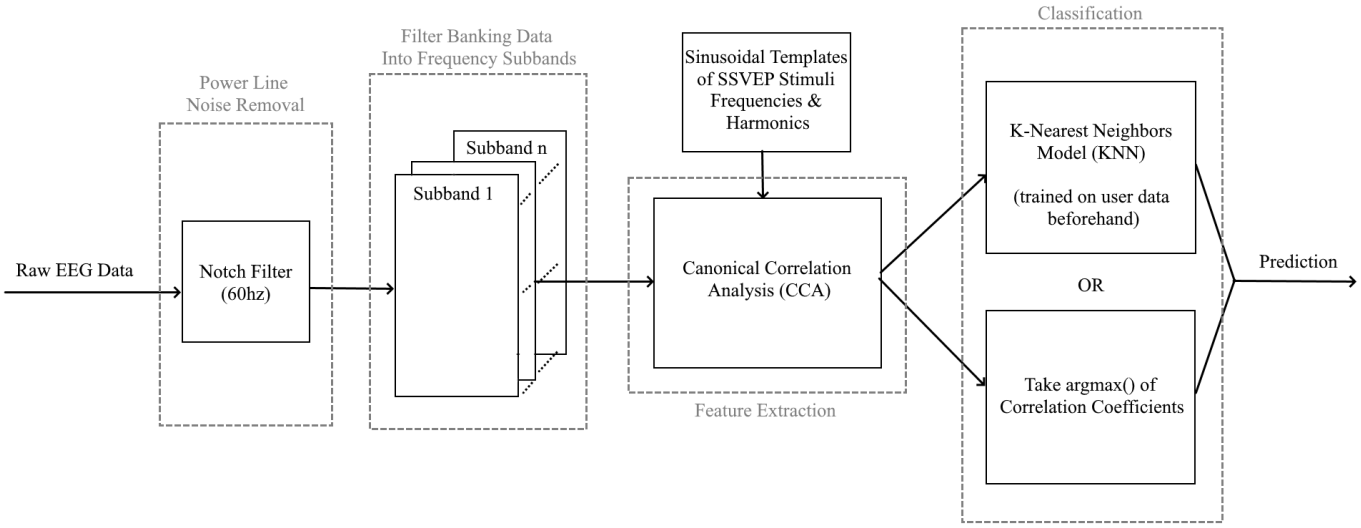

Fig. 5. EEG SSVEP Pipeline Diagram

C. EEG Processing and SSVEP Classification Pipeline

The pipeline ingests raw EEG data at 250 hz. A notch filter is applied at 60 hz to remove power line noise [12].

For further preprocessing and feature extraction we use Filterbank Canonical Correlation Analysis (FBCCA). This consists of two steps, with the first step being a filterbank technique that decomposes the signal into 10 frequency subbands following the "M3" subband decomposition from [6].

The second step involves using Canonical Correlation Analysis (CCA). CCA takes in both the processed EEG subbands and the sinusoidal reference templates matching the stimuli base and harmonic frequencies to identify correlation between the EEG signal and the stimuli frequencies [13].

To do this CCA computes pairs of coefficient vectors that correspond to projections on the EEG data and the reference templates [13]. We then compute a correlation statistic between these vectors using Pearson Correlation and create a matrix of correlation statistics between each frequency subband and the reference templates. We weigh the correlation statistics using a dot product with pre-defined weights that weigh lower frequency bands more heavily than higher frequency bands. This weighted correlation statistic matrix contains our features.

For classification, the system can either feed the features into a trained K-Nearest Neighbors (KNN) model (number of neighbors set to four) for stimuli prediction, or simply predict stimuli based on the max correlation statistic between a set of subbands and a reference template. For the rest of the paper, these approaches will be referred to as FBCCAKNN and FBCCA-max respectively.

D. Large Language Model Usage in SSVEP Interface

The figure below Illustrates the most common user flow of our application.

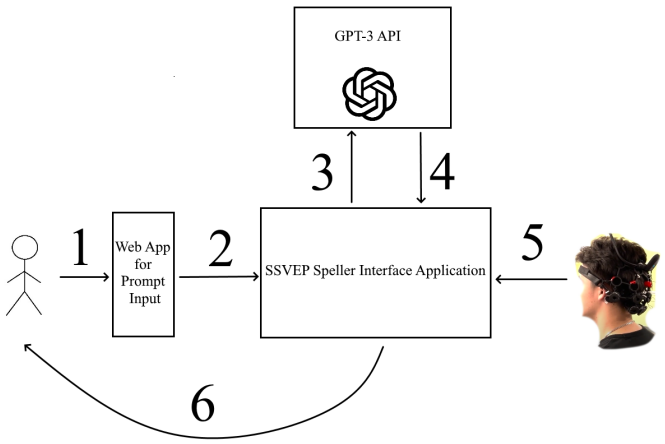

Fig. 6. SSVEP Interface User Flow Diagram

1) Able-bodied user talks to BCI user using speech to text or simply typing in a companion web app.
2) Message is sent to the interface.
3) Interface queries GPT-3 for phrase responses based on input message from 1.
4) GPT-3 returns phrase suggestions.
5) BCI user forms response with phrase suggestions.
6) Text to speech layer reads response aloud to able-bodied user.

For our SSVEP speller, instead of defaulting to characters, our keys defaulted to phrase suggestions obtained using OpenAI's GPT-3 API, more specifically, the davinci model was used for our tasks [14].

The entire SSVEP interface application, from data streaming, to the pipeline, to the graphical user interface was written in Python. This application and its web app companion can

be found on our GitHub repository [15].

III. Results (Preliminary)

The first set of results were found by having two subjects navigate and type messages in our SSVEP interface in two separate sessions, one session using the FBCCAKNN classifier and one using the FBCCA-max classifier. It is important to note that for each subject, the KNN in FBCCAKNN was trained separately on the subjects data alone. The number of total key selections and correct key selections were recorded from each session, and classification accuracy metrics for each session were determined from this.

TABLE I
SSVEP Classification Performance

Subject	Classifier	Selections	Correct Selections	Accuracy
1	FBCCA-max	24	21	87.50%
1	FBCCAKNN	27	23	85.18%
2	FBCCA-max	36	30	83.33%
2	FBCCAKNN	31	28	90.32%

Based on the results from Table 1, it is unclear whether the FBCCAKNN classifier performed better than the unsupervised FBCCA-max approach, more data from more subjects would be needed to prove a significant difference in classification accuracy. Considering FBCCAKNN requires additional data collection and training beforehand, it is safe to say (at least with these early results) that FBCCA-max has a better setup effort to accuracy ratio compared with FBCCAKNN.

The next set of results come from a conversation we had with one subject while they used our SSVEP interface. We spoke back and forth with the subject three times, and they responded to each of our prompts using the phrase suggestions on the SSVEP interface that came from GPT-3.

TABLE II
ITR Performance from Example Conversation using GPT-3 responses (FBCCA-max)

Response	Time to respond (s)	Words	ITR (words/min)
1	30	17	34.00
2	35	10	17.10
3	62	15	14.50
AVG	42.67	14	21.87

Based on the results in Table 2, although preliminary, it is still very significant that on average in this example our system's ITR is over two times that of the highest we could find (10 words a minute [6]). This is very promising for the future of wearable speech BCIs, and demonstrates that Large Language Models can significantly improve ITR for these systems.

IV. Conclusion

To conclude, our team has shown that existing methods in EEG BCI literature along with the incorporation of LLMs work well together, and in some contexts can dramatically improve rate of speech communication. This ultimately brings us one step closer to minimizing the gap between a wearable BCI user's rate of communication and that of an able-bodied individual.

One of the disadvantages of our implementation however was that it did not have any personalization. This meant that it only took into account the prompts it was receiving in making phrase suggestions and it did not take into account environmental, personal, or relationship contexts when forming phrase suggestions, meaning more personalized messages would still take a lot longer to be communicated.

Although conversational context was taken into account, future research should adopt a speech language pathology perspective to work towards quicker and more personalized communication. This could be achieved by implementing an additional context engine, considering environmental and personal context, for example, the relationship the user has with the person they are communicating with.

References

[1] M. T. Medina-Juliá, Á. Fernández-Rodríguez, F. Velasco-Álvarez, and R. Ron-Angevin, "P300-based brain-computer interface speller: Usability evaluation of three speller sizes by severely motor-disabled patients," Frontiers in Human Neuroscience, vol. 14, 2020.

[2] "Electroencephalogram (EEG)," Electroencephalogram (EEG) — Johns Hopkins Medicine, 08-Aug-2021. [Online]. Available: https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/electroencephalogram-eeg. [Accessed: 12-Mar-2023].

[3] M. Li, D. He, C. Li, and S. Qi, "Brain–Computer Interface speller based on steady-state visual evoked potential: A review focusing on the stimulus paradigm and performance," Brain Sciences, vol. 11, no. 4, p. 450, 2021.

[4] N. Shi, L. Wang, Y. Chen, X. Yan, C. Yang, Y. Wang, and X. Gao, "Steady-state visual evoked potential (ssvep)-based brain–computer interface (BCI) of Chinese speller for a patient with amyotrophic lateral sclerosis: A case report," Journal of Neurorestoratology, vol. 8, no. 1, pp. 40–52, 2020.

[5] "Units Converters," unitsconverters.com, 2016. [Online]. Available: https://www.unitsconverters.com/en/Bitpersecond-To-Wordperminute/Unittounit-4798-7602. [Accessed: 12-Mar-2023].

[6] X. Chen, Y. Wang, S. Gao, T.-P. Jung, and X. Gao, "Filter Bank canonical correlation analysis for implementing a high-speed SSVEP-based brain–computer interface," Journal of Neural Engineering, vol. 12, no. 4, p. 046008, 2015.

[7] D. Barnard, "Average speaking rate and words per minute," VirtualSpeech, 08-Nov-2022. [Online]. Available: https://virtualspeech.com/blog/average-speaking-rate-words-per-minute. [Accessed: 12-Mar-2023].

[8] R. Kuś, A. Duszyk, P. Milanowski, M. Łabecki, M. Bierzyńska, Z. Radzikowska, M. Michalska, J. Żygierewicz, P. Suffczyński, and J. Durka, "On the Quantification of SSVEP Frequency Responses in Human EEG in Realistic BCI Conditions," PloS one, vol. 8, no. 10, Oct. 2013.

[9] "Wireless 8-channel EEG headset," Unicorn Hybrid Black, 12-Jan-2023. [Online]. Available: https://www.unicorn-bi.com/. [Accessed: 12-Mar-2023].

[10] "Ultracortex 'Mark IV' EEG headset," OpenBCI Online Store. [Online]. Available: https://shop.openbci.com/products/ultracortex-mark-iv. [Accessed: 12-Mar-2023].

[11] F. Lotte, L. Bougrain, and M. Clerc, "Electroencephalography (EEG)-based brain-computer interfaces," Wiley Encyclopedia of Electrical and Electronics Engineering, pp. 1–20, 2015.

[12] S. O. Gilani, Y. Ilyas, and M. Jamil, "Power line noise removal from ECG signal using notch, band stop and adaptive filters," 2018 International Conference on Electronics, Information, and Communication (ICEIC), 2018.

[13] M. Nakanishi, Y. Wang, Y.-T. Wang, and T.-P. Jung, "A comparison study of canonical correlation analysis based methods for detecting steady-state visual evoked potentials," PLOS ONE. [Online]. Available: https://doi.org/10.1371

[14] OpenAI API. [Online]. Available: https://platform.openai.com/docs/models/gpt-3. [Accessed: 12-Mar-2023].
[15] "Watolink," GitHub. [Online]. Available: https://github.com/WATOLINK. [Accessed: 12-Mar-2023].

Creating a Corn Yield Estimator: A Deep Learning Approach via UAV Imagery

Sanindie Silva
Queen's University
18asns@queensu.ca

David Courtis
Queen's University
20dhc@queensu.ca

Jax Hodgkinson
Queen's University
20jth7@queensu.ca

Severn Lortie
Queen's University
severn.lortie@queensu.ca

***Abstract*—This project aims to develop a fast, accurate, and cost-effective tool for yield prediction of maize plants using computer vision models and deep learning. Two models trained on a large dataset of UAV imagery are combined together to detect the number of maize plants (TasselNetV2+) and detect any unhealthy crops (CORN-DOC). The algorithm can be used by farmers, and other stakeholders in the agriculture industry to make informed decisions about their crops. Other features have been incorporated into our algorithm, like the average amount of ears of corn per stalk and the average weight per ear. These features will be combined together into a website, where farmers can upload images of their fields taken by UAVs and get an estimated yield that will be calculated using distributed computing power. Overall, this project can help to improve farm management practices and maximize crop yields, ultimately leading to improved food security and economic benefits for farmers and communities.**

I. Introduction

A. Motivation

The agricultural sector has long been in need of reliable and efficient solutions for accurately counting crop yield. Counting yield in farming is crucial as it helps farmers monitor the growth status of their crops, make informed decisions about crop management practices, and estimate crop yield. Traditional manual observation is highly labour-intensive and inefficient, while existing automated plant counting systems are limited to controlled lab environments and specific scenarios to which conditions are not transferable. However, there has been a growing interest in developing automated plant counting tools based on digital imagery, and many of these tools are motivated by the success of deep convolutional neural networks (CNNs). The emergence of deep learning-based computer vision technology, combined with the prevalence of digital cameras and increased computational resources, has opened up new possibilities for practical applications of plant counting in the field. The existing plant counting networks have reported remarkable performances already.

Our project aims to build on the latest research in computer vision and deep learning surrounding maize to develop a fast and accurate plant counting application that farmers can leverage in real time worldwide. Specifically, we are iterating on an existing algorithm, TasselNet, which is trained on a large dataset of UAV imagery to detect maize tassels in the field. Maize tassels are the male part of the plant that grows when the plant reaches maturity and is ready to be harvested. They are thin and yellow, which helps models to recognize tassels from the rest of the green leafy plant. The ability to accurately count maize tassels is essential for monitoring the growth status of maize plants, as it provides important information about the plant's growth stage and potential yield. This information is critical for making informed decisions about crop management and maximizing crop yields, which can significantly impact food security and economic benefits for farmers and communities

B. Related Works

In recent years, advancements in computer vision, machine learning, and deep learning have enabled precision agriculture to forecast crop yields accurately. This paper focuses on exploring the different techniques used to forecast crop yields. In particular, we examine the following three studies and compare their methodologies: "TasselNet: counting maize tassels in the wild via local counts regression network," "TasselNetV2+: A Fast Implementation for High-Throughput Plant Counting From High-Resolution RGB Imagery," and "Simultaneous corn and soybean yield prediction from remote sensing data using Deep TransferLearning."

The article "TasselNet: counting maize tassels in the wild via local counts regression network" presents a method for automated counting of maize tassels using images captured in the field. The method uses a deep learning architecture called TasselNet, which combines a convolutional neural network with a local counts regression network. The TasselNet is trained to detect and count the number of tassels in each image. The authors evaluated the performance of TasselNet on a large dataset of field images and compared it with other methods. They found that TasselNet achieved high accuracy and outperformed other state-of-the-art methods. This model sets a good baseline for future models to be based on, however, this article lacks a detailed explanation of how the TasselNet architecture was designed or optimized, making it difficult to reproduce or extend the method [4].

After a few iterations, the authors of the Tasselnet article released another iteration called TasselNetV2 that built off of the original model. The article "TasselNetV2+: A Fast Implementation for High-Throughput Plant Counting From High-Resolution RGB Imagery" stated that there was an improvement to TasselNetV2, called TasselNetV2+. Compared to TasselNetV2, TasselNetV2+ can achieve around 30 frames

per second on images with a resolution of 1980×1080 while retaining the same level of accuracy as TasselNetV2. TasselNetV2+ can be faster than its predecessor because it focuses on object counting, not detection. Unlike object detection, object counting requires only dotted annotations, simplifying the network architecture and reducing the cost of annotation within a lightweight computational requirement. Furthermore, while TasselNetV2 was focused on wheat counting, TasselNetV2+ is trained on maize data in addition to other plants [3]. There is another iteration called TasselNetV3, however, the documentation was lacking and was unclear on how to replicate the model [5].

The article "Simultaneous corn and soybean yield prediction from remote sensing data using Deep Transfer Learning", published in the journal Scientific Reports, presents a computer vision-based algorithm to detect and count maize tassels from high-resolution RGB imagery captured by Unmanned Aerial Vehicles (UAVs) for high-throughput maize counting in the field. The proposed method uses a deep convolutional neural network and segmentation to accurately detect maize tassels with a YieldNet model. The algorithm is optimized for real-time performance, but it was run on a high-performance computing cluster, which is not accessible to potential users of the technology. Furthermore, while it can be used to estimate maize yield in the field, the algorithm was tested on images captured under controlled environmental conditions in a greenhouse, but the performance may be impacted in real-world settings where lighting and other environmental factors are more variable [2].

In conclusion, the three studies reviewed in this paper demonstrate the potential for deep learning-based methods to count and forecast crop yields in agriculture accurately. TasselNet and TasselNetV2 provide promising solutions for the automated counting of maize tassels with high accuracy, and TasselNetV2+ offers a faster and more efficient version of its predecessor. The algorithm proposed in the third study offers a real-time, high-throughput solution for maize counting, but its effectiveness in more variable environmental conditions is yet to be seen. While these studies show promising results, there is yet to be a fast and accurate application trained on field data that can be used right now. There are steps that can be taken forward to ensure the practicality and applicability of these methods in real-world agricultural settings.

C. Problem Definition

Manual plant counting, currently used in most regions of the world, is a subjective, labour-intensive, error-prone and inefficient task due to human error and fatigue [7]. While attempts have been made to automate this task over the past decades, achieving this goal has been challenging due to the varieties of plants and intrinsic/extrinsic variations in reality. Furthermore, automated plant counting systems developed in the past are often limited to controlled environments or specific scenarios [4]. In this work, we aim to provide an efficient and effective tool for maize counting from high-resolution RGB imagery that farmers in Kenya can use.

We intend to implement a fast, accurate and cost-effective tool for farmers that can be easily deployed in the field. We are leveraging a fast version of the state-of-the-art plant counting model called TasselNetV2+, which is trained on a large dataset of UAV imagery and uses computer vision techniques to detect maize tassels in the images. Compared to previous iterations TasselNetV2+ can achieve around 30 frames per second on images with a resolution of 1980×1080 while retaining the same level of accuracy. [3]. TasselNetV2+ can be faster than its predecessor because it focuses on object counting, not detection. Other commonly used models, such as Faster-RCNN, are even further computationally expensive, especially for high-resolution imagery used in modern plant phenotyping platforms [1].

Furthermore, this team plans to include other features into our algorithm to better predict maize fields yield, like the average amount of ears of corn per stalk and leaf health. These features will be combined into a website, where farmers can upload images of their fields taken by UAVs and get an estimated yield that will be calculated using distributed computing power. The website will enable farmers and other stakeholders in the agriculture industry to make informed decisions about their crops.

II. Methodology

Our project is a collaboration with Nikola Energy, a Kenyan agriculture company that guided the creation of our yield estimator. We started by determining which data types would be useful for our prediction model and would have a relatively easy collection process. We considered stationary aerial images, satellite imagery and handheld cameras, but ultimately we concluded that drone-based imagery would be the most reliable and efficient method for our use case. Drone imagery or Unmanned Aerial Vehicle (UAV) imagery provides a stable platform for image collection and offers more metadata to determine the area of the field in the image. Compared to handheld cameras, drone shots are more consistent and, therefore, easier to train a model on, while providing better visibility than stationary aerial and satellite images. Having determined the type of data we would use, we started examining similar projects and models, as mentioned in the related works section, to find suitable labelled data.

A. TasselNet

We selected an MTC-UAV dataset [5]. that was used to train another version of TasselNet, which included 306 UAV images from 400 maize cultivars and 70 870 manually point-labelled instances. The images were captured from a height of 12.5 meters with a resolution of 5472 × 3648 in varying in-field environments. Alternative datasets such as MTC [4]. and YieldNet sets [2] were explored; however, due to the resolution and height requirements of drone imagery, the MTC-UAV dataset best represented our application.

The first step was using the MTC-UAV images on YieldNet [2], Faster-RCNN [1] and TasselNetV2 [7] to determine the model's initial performance. The models were compared based

on their RSE and R2 scores that measured the accuracy of the model, this gave us a consistent way to compare models and act as a performance benchmark for accuracy and speed. Some models that were designed for handheld or satellite imagery had their accuracy suffer when losing information like multispectral bands or images over several weeks. We also tracked the computational requirements of each model, since we wanted to ensure our final project was both cost-efficient and lightweight. While the training process did not need to be optimized, since it would be done rarely and on specialized hardware, the evaluation model needed to be fast enough to be initialized quickly or lightweight enough to be always available.

After rebuilding the initially available models, we determined that TasselNetV2 was the fastest and most accurate model tested. With further research, our goal was to expand on the initial model and incorporate features of TasselNetV2+ and TasselNetV3 to get a faster speed and higher accuracy, respectively. Ultimately we successfully recreated TasselNetV2+ and tested it on various combinations of layers and preprocessing techniques to optimize accuracy and speed.

B. CORN-DOC

Furthermore, we were determined to incorporate a "health check" to identify diseased crops, so we developed a second model that would work in conjunction with TasselNetV2+ to create the final yield estimate. We used a dataset from OSF [6] that contains 18,222 field images, and 105,735 annotations for Northern Leaf Blight (NLB). NLB is a disease caused by the fungus *Setosphaeria Turcica*, which is one of the most "economically damaging maize diseases" [6]. The images were taken from varying height levels and image quality.

The NLB dataset was used to create a convolution neural network called "CORN-DOC (CORN - Disease Oberservation and Counter)." CORN-DOC splits the input images into patches of 100x100 and sends them to a simple CNN model with fast loading of weights that predicts the confidence of a disease in a certain patch. The model can also generate an annotated image of the disease locations.

The TasselNetV2+ model would perform the initial estimate of the number of maize plants, and our second model would detect for any Northern Leaf Blight in the UAV images of the maize crops. If any diseased crops were detected, the estimate would be adjusted accordingly.

C. User Interface

We also incorporated user input into the estimator to provide farmers with greater flexibility in the yield calculation algorithm. The frontend of the estimator would be designed to be easy to use and would include buttons for further configurations. For example, the TasselNet model can only count the number of maize plants, and our estimator was working with the assumption that "For every maize plant, there would be one ear on the stalk on average," however, we included an input field for users to adjust the average number of ears per stalk to better reflect their past crop yield. Similarly, we included an input for users to adjust the average weight of maize per plant, where the default is 340g per maize ear. The application would provide the estimated total amount of maize ears and the estimated weight of the total maize predicted based on the default averages, which could also be adjusted by user input automatically within the frontend.

D. Resources

The models discussed in this paper were trained and tested on a GeForce RTXTM 3070 Ti graphics card. We intend to integrate distributed computing into the final pipeline to distribute the computing power needed for the CORN-DOC model. Since TasselNetV2+ is a custom model, at this time, we cannot distribute its compute, however, to overcome this challenge, we specifically chose a lightweight and fast model that is more portable and useable on less powerful resources.

Overall, the plan was to create a comprehensive yield estimator for maize fields using UAV imagery. While we realize that we are taking strong liberties with the yield calculation algorithm, we have taken steps to incorporate user input and provide flexibility in the final algorithm to better reflect a farmer's specific conditions. With the combination of these models, the flexibility of the final yield algorithm and an easy-to-use frontend, we hope to give farmers the ability to quickly and accurately predict the yield for their fields.

III. Results

With the UAV dataset, our implementation of the TasselNetV2+ model showed good accuracy with R2 values of up to 95% and MAE values of 17.3. This is a very small error for crop counting and yield prediction and shows that our model will absolutely be useful for counting corn tassels using UAV images.

TABLE I
COMPARING THE ACCURACY OF THE MODELS REPRODUCED

Model	MAE	MSE	RMAE	RMSE	R2
YieldNet	90.569	12,364	–	–	0.086
Faster-RCNN	5.132	7.70	–	9.10	0.8727
TasselNetV2+ on MTC	5.090	9.06	–	9.00	0.8800
TasselNetv2+ on UAV	17.36	24.73	10.71	16.34	0.9578

Alongside this main counting model, our disease identification model, CORN-DOC, has also had high accuracy with a R2 value of 91.8% and a MSE value of 0.0239. Figure 1 shows an example performance of the CORN-DOC model. Combining these models together gives a good estimate for the final yield of the field and means farmers will have a fast and accurate way of determining how their crops are doing.

TABLE II
CORN-DOC MODEL ACCURACY

Model	MAE	MSE	RMAE	RMSE	R2
Disease Model	0.0609	0.0163	0.2432	0.1127	0.9517

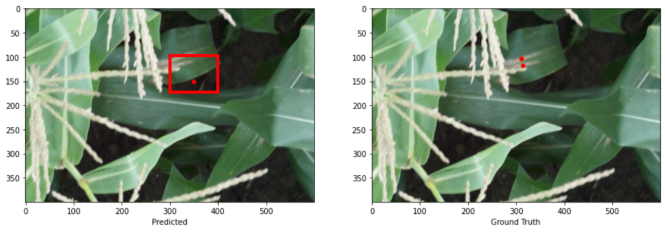

Fig. 1. Example of CORN-DOC's prediction on a random image.

IV. Conclusion

A. Ethical Dilemmas

The team identified three key ethical dilemmas during the creation of the model. They are model transparency, accessibility, and side effects. Model transparency means providing a user with some level of insight as to how the model came to the conclusions it did. Model accessibility means the ease with which our target users, farmers, can deploy the AI system. Lastly, the side effects of the model are any unintended consequences brought about by its intended and unintended uses.

Model transparency was a key concern. Letting users understand the system's operation will allow it to be a trustworthy source of information. This is especially important when its output, yield counts, is mission critical for most farmers. To achieve this, we provided two services. The first is a visualization of the count density map for each input image. This gives a graphical representation of how the model sees the image. Users can verify the operation of the model using these. The second service is the ability to tune how the system handles diseased corn when estimating counts. Users can edit the model's constants to account for diseased/unviable stocks. Overall, these services help to address concerns of model transparency by allowing for inspection and alteration of its decision-making.

Model accessibility was determined to be mainly influenced by cost and ease of use for this project. The cost would come from the compute needed for inferencing and training. We mitigated this by hosting the model on Distributive's compute platform. This platform allows for cheap compute by distributing the work to many anonymous workers. The cost per operation is vastly less than competitors such as Amazon or Google. Ease of use was achieved by designing an intuitive GUI for interaction with the system. Users can easily upload their crop images and retrieve results without any knowledge of AI or distributed computing required.

Unintended consequences are those which might go overlooked during the production of the product but become problematic when it is deployed. For this project, a large consideration was how the target users would interpret the results given by the model. While our counts are highly accurate (within 95% of ground truth), they are not a complete estimate of crop yield. Maize plants suffer from various diseases, such as Gray Leaf Spot, Northern Corn Leaf Blight, and southern rust. These diseases impact the quality of maize and, in some cases, render all of it from a region of the crop unusable. If the model did not account for disease and instead reported just the corn count, users may incorrectly assume that yield is better than reality. To help combat this, a secondary disease estimation network was added to enable adjustments for potential crop maladies. This model reduces the overall count by an appropriate factor to compensate for NLB, a problematic disease present in the region of deployment, however, further efforts can be made to account for other diseases.

B. Future Recommendations

While the team behind this project is very proud of the work we have managed to accomplish, we realize that there is room for improvement. As mentioned earlier, there are more advanced models within this area of research, namely TasselNetV3. While the model we used, TasselNetV2+, is very accurate and useful for our purposes, it can be improved upon, as a result, the next step would be to try to recreate TasselNetV3 in python - due to this team's lack of experience and lack of clear documentation within the TasselNetV3 article, we were unable to properly implement the changes between TasselNetV2 and TasselNetV3 into our model.

Furthermore, our estimator only monitors crops for Northern Corn Leaf Blight, while several other diseases can impact maize yield. Future iterations can find appropriate data and include models that would identify other types of diseases. Another way to improve the yield estimator would be to include other types of data concerning agriculture, like soil quality or ph levels. Unfortunately, although the factors for proper agriculture practices are well known, there is not a readily available dataset that can be leveraged. If working in conjunction with agricultural companies/groups, future research can work towards collecting other appropriate agricultural data to create models from.

One of the biggest drawbacks of our estimator is that we use approximate averages within our yield estimator's algorithm, which could lead to inaccurate estimates. Our solution was to include variable values (average ears/stalk, average weight of cob) with a default value that can be changed by user input. However, this problem could be addressed by creating a model based on landscape images of maize fields, counting the number of maize on an outer row, and calculating an average estimate of the number of ears of maize per stalk. The main drawback to this plan would be finding a suitable dataset. Nonetheless, these suggestions would build upon our model to create a better tool for farmers to use to their benefit.

Future iterations can also look into the detection and spread of NLB within fields. Due to our model's efficiency, we can deploy low-cost, recurring services to potential users, allowing for consistent and rapid monitoring of crop yield and growth. Due to the high speed and low cost, an avenue for expansion is thereby found within the potential for early disease detection and traceability by comparing disease detection data between scans. Because of the high prevalence of diseases within maize

fields, it is essential to detect and prevent the spread of disease as early as possible.

C. Closing Remarks

Overall, this project aims to address the challenges associated with manual plant counting and provide a fast, accurate, and cost-effective tool for maize counting in the field. By leveraging the latest computer vision and deep learning research, we aim to develop an algorithm that can detect and count maize tassels in real time from high-resolution RGB imagery captured by UAVs. This tool will enable farmers to make informed decisions about crop management and maximize crop yields, which can have a significant impact on food security and economic benefits for farmers and communities.

In terms of the next steps, our team plans to further optimize the algorithm for real-time performance using Distributed Compute Protocol and finetuning other features integrations in the tool that can provide more accurate yield predictions, such as the average amount of ears of corn per stalk and leaf health. We also plan to further develop our user-friendly website where farmers can easily upload images of their fields and receive estimated yield calculations using distributed computing power. Challenges that remain include addressing variations in the field-based environment, such as illumination changes and occlusions, and ensuring that the tool can accurately count maize tassels across different cultivars. Nevertheless, we are excited about this tool's potential impact on the agricultural sector and look forward to continuing its development.

ACKNOWLEDGEMENT

This team would like to give special thanks to Nikola Energy and Distributed Compute Labs for their continued support and collaboration within this project.

REFERENCES

[1] Alzadjali, A., Alali, M. H., Veeranampalayam Sivakumar, A. N., Deogun, J. S., Scott, S., Schnable, J. C., &; Shi, Y. (2021). Maize tassel detection from UAV imagery using Deep Learning. Frontiers in Robotics and AI, 8. https://doi.org/10.3389/frobt.2021.600410

[2] Khaki, S., Pham, H., &; Wang, L. (2021). Simultaneous corn and soybean yield prediction from remote sensing data using Deep Transfer Learning. Scientific Reports, 11(1). https://doi.org/10.1038/s41598-021-89779-z

[3] Lu, H., &; Cao, Z. (2020). TASSELNETV2+: A fast implementation for high-throughput plant counting from high-resolution RGB imagery. Frontiers in Plant Science, 11. https://doi.org/10.3389/fpls.2020.541960

[4] Lu, H., Cao, Z., Xiao, Y., Zhuang, B., &; Shen, C. (2017). TasselNet: Counting maize tassels in the wild via local counts regression network. Plant Methods, 13(1). https://doi.org/10.1186/s13007-017-0224-0

[5] Lu, H., Liu, L., Li, Y.-N., Zhao, X.-M., Wang, X.-Q., &; Cao, Z.-G. (2022). Tasselnetv3: Explainable plant counting with guided upsampling and background suppression. IEEE Transactions on Geoscience and Remote Sensing, 60, 1–15. https://doi.org/10.1109/tgrs.2021.3058962

[6] Wiesner-Hanks, T., & Brahimi, M. (2019, February 11). Image set for deep learning: Field images of maize annotated with disease symptoms. Retrieved from osf.io/p67rz

[7] Xiong, H., Cao, Z., Lu, H., Madec, S., Liu, L., &; Shen, C. (2019). TASSELNETV2: In-field counting of wheat spikes with context-augmented local regression networks. Plant Methods, 15(1). https://doi.org/10.1186/s13007-019-0537-2

APPENDIX

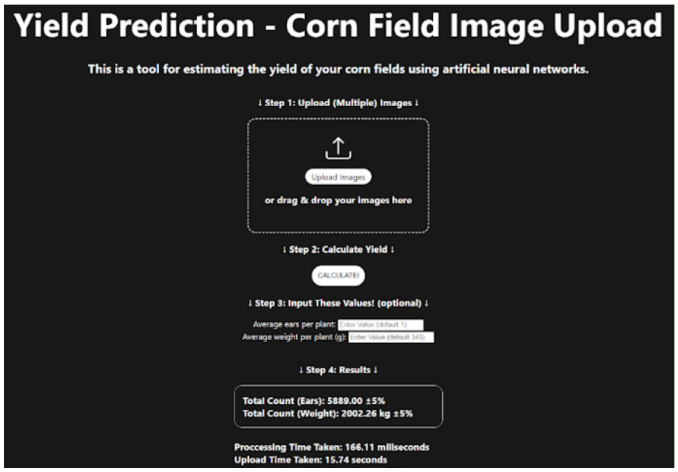

Fig. 2. Example of the tool's frontend.

Improving Public Transit Systems through Clustering and Polynomial Regression

George Trieu
Queen's University
g.trieu@queensu.ca

Pavle Ilic
Queen's University
20pi@queensu.ca

Cahal Ng
Queen's University
20cn16@queensu.ca

Owen Rocchi
Queen's University
19omr6@queensu.ca

Duncan Cheng
Queen's University
17jc66@queensu.ca

***Abstract*—As urban mass transportation systems worldwide evolve, they must adapt to the ever-changing needs of their residents. This paper explores a machine-learning approach using clustering and polynomial regression. This model fits transportation corridors and hubs to optimize the population served using points of interest and hot spots. Scoring of station and line candidates is performed through a summation of Gaussian distributions between the candidate and each point of interest. A novel approach to generating transportation interchanges through heuristics is also explored.**

I. Introduction

As urban centre populations continue to develop into the 21st century, it is essential to design efficient and cost-effective mass transportation. With differences in geography and routines being local to each region around the world, there is no universal system for planning public transportation. Urban and city planning today leverages highly advanced tools, such as Geographic Information Systems (GIS) to develop transit systems. Often, this process takes years to plan and decades to execute, all while urban compositions and the needs of the public continue to evolve. Building transit for the future involves designing for people today and tomorrow, not from the past. The most significant reason car usage continues to grow in large metropolitan areas is because of the inconvenience associated with using mass transit options – a symptom of poor and outdated public transit planning. This paper aims to explore different machine learning methods to bring mass transit proposals and implementation plans to governing bodies faster. Options such as heavy rail and light rail are the focus of this paper.

A. Motivation

Scientists in Japan experimented with a slime mould simulating the Tokyo railway network using oat flakes and a single-celled mould called Physarum polycephalum [1]. The connections the mould generated mimicked the current Tokyo transit system, a system designed by thousands of engineers.

With the vast amount of geographical data that is publicly available, machine learning could be applied to these large data sets to generate optimized solutions for urban planners. An application that can give realistic solutions to rapid transit lines in any city is the focus of this experiment. As the need for better and more efficient transit grows around the world, it is imperative to make use of machine learning.

Leveraging machine learning tools to design efficient urban mass transit routes can be much more powerful than manual planning. Human-planned public transit tends to stick to pre-existing trends – whether that may be arterial roads, railroad lines, or geography. Machine learning algorithms rely on a higher dimensional feature set that is hard for humans to perceive, resulting in far more diverse and unique ideas.

B. Related Works

There have been numerous works of literature related to transit route planning. An example is "Improving Transit Accessibility with Machine Learning" by Google AI Blog, where the prediction of transit wait times utilizes real-time traffic forecasts and data on bus routes and stops [2]. The model is split into a sequence of timeline units per bus, with its wait duration independently forecasted [2]. Additionally, a guide on transit station site selection to address good practices in the model city of Eau Claire, Wisconsin, USA helps outline the importance of amenities and points of interest around a station [3]. This guide supports a lot of the planning of Eau Claire's transit system through the analysis of other transit systems across the United States [3].

C. Problem Definition

With the lack of automation and the use of artificial intelligence in transit planning, the solution is to build a model to design an optimized subway system for any city in the world. This provides a tool for urban and city planners to make informed decisions when designing and implementing a new subway system. It will also give members of the public an opportunity to compare any current subway system with the most efficient and cost-effective version. The output of the model is a system to improve the subway infrastructure of cities, so a greater percentage of the population can be served through transit. To ensure the designed system is convenient for passengers, interchanges must be implemented to facilitate a smooth and easy transition across subway lines.

Overall, the results of this experiment are measured by an arbitrary metric that represents the number of people on average each station serves. In particular, the City of Toronto is used as the pilot city for this experiment. The system generated from the model is compared to the current Toronto Transit Commission (TTC) system to determine the effectiveness of the model.

II. Methodology

The data sets are created through OpenStreetMap API requests. In particular, Points of Interest (POI) like community centres, libraries, and restaurants are found for a given city and saved to individual CSV files to be used by the model. The attributes are filtered to contain the latitude, longitude, and name of each POI, with all other attributes being dropped.

A scoring function assigns coordinates a measure of usefulness. The scoring function is based on a Gaussian Distribution:

$$f(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d}{\sigma})^2}$$

where d is the Haversine distance between a point and a POI, and σ is the standard deviation. The function $f(d)$ generates high scores for points close to the POI and generates scores approaching zero as the distance d increases. This scoring function is applied to a grid search function to give scores to each grid point throughout a city when compared with the locations of POIs. Through hyperparameter tuning, a standard deviation of $\sigma = 0.9$ is used for this experiment.

To determine the high-scoring locations throughout an urban centre, a grid search is performed. This involves dividing the city into an N x N grid. Each point in the grid is scored based on the distance from itself and the locations of each POI using the following equation:

$$s_{i,j} = \sum_{k=0}^{N} f(d)$$

where $s_{i,j}$ is the score at (i, j), N is the number of POIs, and $f(d)$ is the scoring function. The plot of generated grid points for the City of Toronto is shown in Fig. 1.

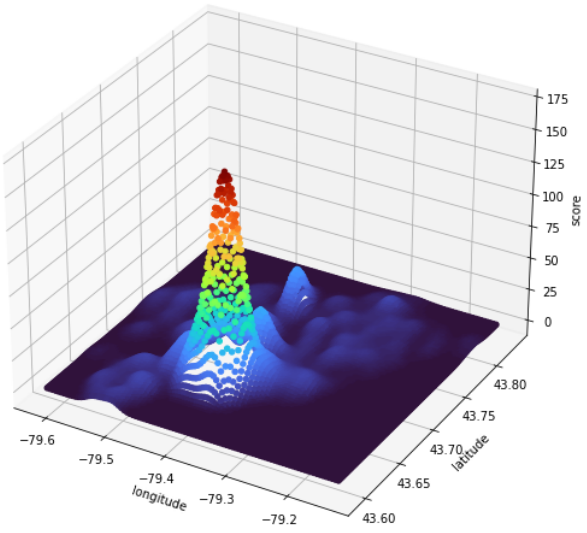

Fig. 1. Scored grid points from the City of Toronto

In any mass urban transportation system, lines in a system should be dispersed throughout the urban area, with an additional level of concentration in the densest part (often the downtown core). Dispersion of transit lines is achieved through clustering. A modified k-means algorithm is used, that considers the scores of each point x_i as a weight w_i in $\mathbb{R}^2$ space [4]:

$$\text{centroid} = \sum_{i=0}^{N*N} \min_{\mu_j \in D} (w_i * ||x_i - \mu_j||^2)$$

where μ_j is the mean of the samples in one of the clusters in the set D. The weighted k-means algorithm converges on a solution where cluster centres are distributed densely in high-score areas, and more sparsely distributed in low-score areas. This is synonymous with having more stations and lines in downtown areas, and fewer stations and lines in the suburbs. The clusters represent the neighbourhoods each line will go through. This implies the number of lines in the system is equal to the number of clusters. The clusters are numbered at random from 0 to C - 1, where C is the number of clusters. Sorting these clusters by average score is important for generating interchanges. After sorting, cluster 0 has points that make up the highest average score, cluster 1 has the next highest, etc. The clusters for $C = 6$ in the City of Toronto are shown in Fig. 2.

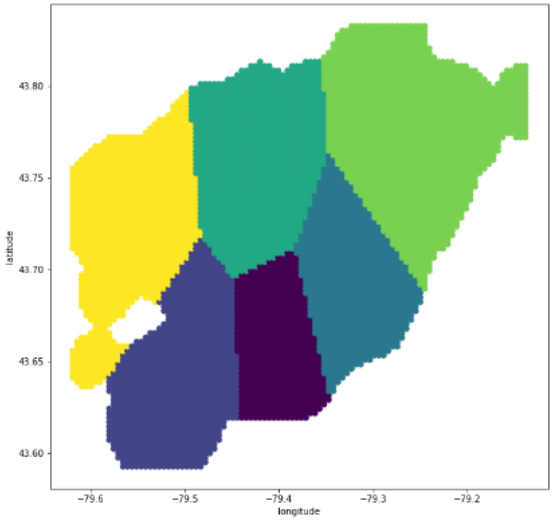

Fig. 2. Clustered grid points from the City of Toronto

After clustering, fitting transit lines is the next step in developing a transit system. A plausible approach is to perform weighted polynomial linear regression independently on each cluster. Weighted polynomial regression is chosen due to the use of weighted grid points. An equation for the matrix calculation of weighted polynomial regression is shown below where m is the degree of the polynomial and n is the number of known data points. It is determined by creating a residual function, summing the squares of the residual, forming a

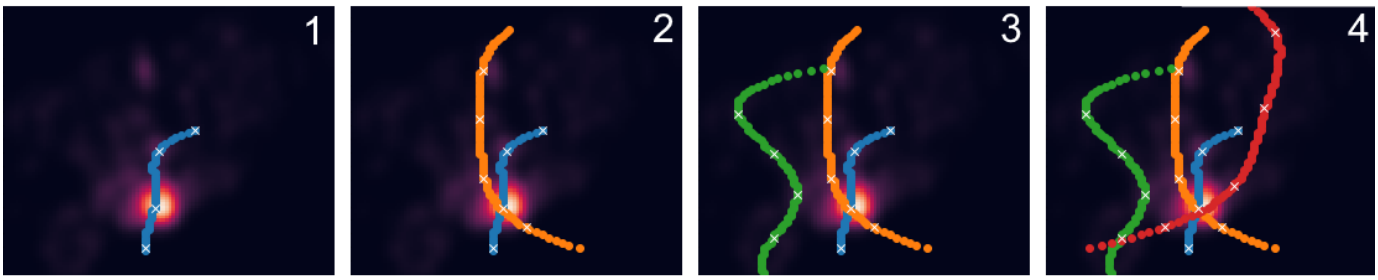

Fig. 3. Line Progression with C = 4 in the City of Toronto

parabola, and determining the coefficients β_i of the parabola [5]:

$$\begin{bmatrix} \sum_{i=0}^{n} w_i & \cdots & \sum_{i=0}^{n} w_i x_i^m \\ \sum_{i=0}^{n} w_i x_i & \cdots & \sum_{i=0}^{n} w_i x_i^{m+1} \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{n} w_i x_i^m & \cdots & \sum_{i=0}^{n} w_i x_i^{2m} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{n} w_i y_i \\ \sum_{i=0}^{n} w_i x_i y_i \\ \vdots \\ \sum_{i=0}^{n} w_i x_i^m * y_i \end{bmatrix}$$

Polynomial regression is used instead of linear regression as it allows for better coverage of the grid points compared to a straight line. A polynomial of degree 7 is used for this experiment to maximize coverage without overfitting. Additionally, the relationship between latitude/longitude and score is not linear in each cluster. The points of the regression lines are snapped onto the existing grid to simplify work with the data in later steps.

Interchanges are an important feature in designing mass public transit systems. By performing polynomial regression on each cluster separately, some unintended consequences are introduced. The resulting system of lines does not overlap as each line is designed to optimize for its constituent grid points (i.e. the set of constituents for each cluster is disjoint). This implies there are no interchanges in the aforementioned system. An iterative approach to generating lines is considered instead. Each line is generated successively from previous lines and utilizes information from previously generated lines. Line generation starts from cluster 0 - the cluster with the highest average score. Potential interchanges or interchange candidates are identified on each line generation through

$$I_l = \underset{i \in L_l}{\operatorname{argmax}}(s_i)$$

where I_l is the interchange candidate for line l, L_l is the set of points in line l, and s_i is the score for the i^{th} point on the line. The score of the grid point associated with I_l is multiplied by a factor α such that the new score s is more favourable to the regression of the next line. The interchange candidates also decay after each iteration. This encourages the generation of future lines to consider other interchanges and not just the first interchange. After each iteration, the score of each interchange candidate is reduced by a decay factor ρ, such that the new score for the i^{th} interchange candidate is $s_i(1-\rho)$. This results in the score of the i^{th} interchange candidate after k iterations to be

$$s_i^{(k)} = \alpha s_i (1-\rho)^k$$

The modified grid point $V_i^{(k)}$ associated with each score at each iteration $s_i^{(k)}$ is added to the set of points for all future lines, i.e.

$$\forall k \in \{0, ..., C-1\}, L_{l+k} = L_{l+k} \cup \{V_0^{(k)}, ..., V_i^{(k)}\}$$

An example of this iterative line generation process for the City of Toronto is shown in Fig. 3. The interchange candidates are represented in the figures by a white x.

To generate stations, the N_s highest-scoring locations on a line are found, and the distance between them is calculated. A station is chosen if the distance between two locations is greater than the distance parameter. The N_s value is a parameter that can be tuned, resulting in a line having at most N_s stations. This method does not guarantee that there will be exactly N_s stations per line as it may not be a large enough distance apart from other high-scoring locations.

III. Results

As stated in the problem definition an arbitrary metric that represents the number of people served on average per station is used to compare the current transit system with the one generated by the model.

The metric is based on iterating on all stations of each line and averaging all the scores of the grid points that lie R radius from the station.

$$s_{\text{station}} = \frac{\sum_{s \in S} s}{||S||}, S = \{s_{i,j} \mid i, j \in \mathbb{Z}, \sqrt{i^2 + j^2} \leq R\}$$

The score of the system is simply the mean score of all the station scores.

Fig. 4 shows the system generated by the model for the City of Toronto, consisting of four lines.

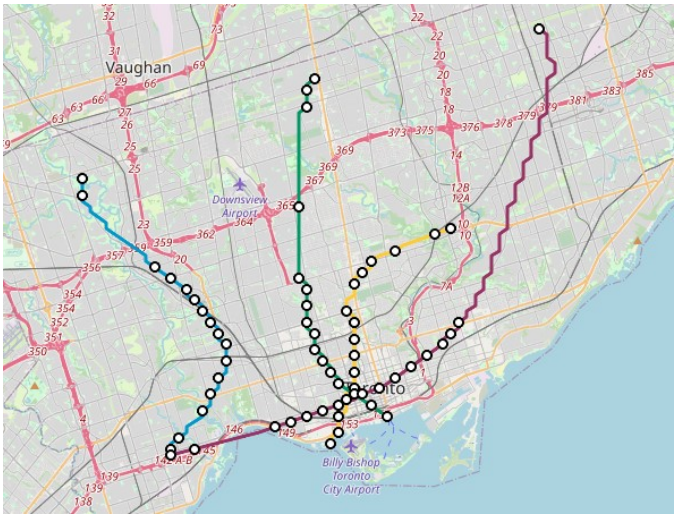

Fig. 4. Generated System for the City of Toronto, $C = 4$

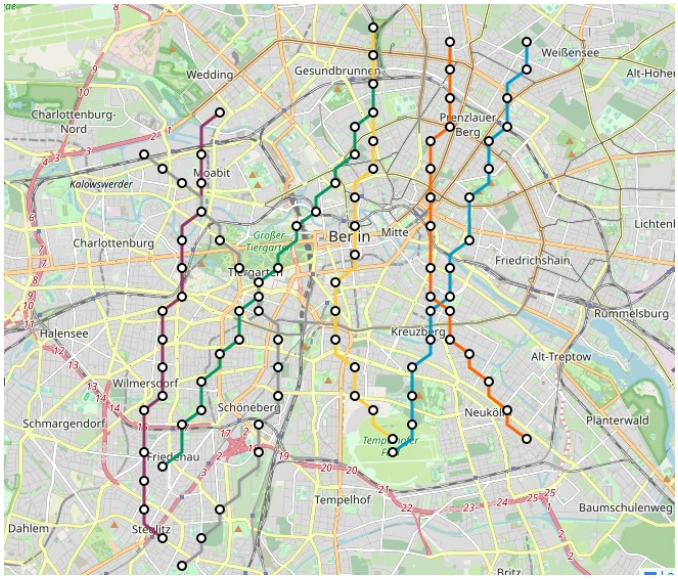

Fig. 5. Generated System for the City of Berlin, $C = 6$

To ensure an accurate comparison between the model and the current TTC system the results were calculated with $C = 4$ as the TTC consists of 4 subway lines. The model uses $R = 1$, representing the distance customers would walk to use a transit station. The current TTC system with 65 stations scored 44.15 using the system scoring metric. Table I shows the system scores generated by the model for Toronto, where N_s represents the number of stations per line. The score differs based on how many stations are chosen per line. A simplification is made where each line is chosen to have the same number of stations. The exception is cases where the line can not support N_s stations due to its length, resulting in fewer stations than N_s. For $N_s \leq 15$, the transit system generated from this experiment outperforms the existing TTC system.

TABLE I
RESULTS FOR GENERATED TRANSIT SYSTEMS FOR THE CITY OF TORONTO, $C = 4$

N_s	Score	N_s	Score	N_s	Score
8	69.75	12	53.40	16	42.83
9	64.70	13	50.30	17	40.91
10	60.46	14	47.64	18	39.13
11	56.68	15	45.07	19	37.48

A transit system for Berlin with $C = 6$ is generated as shown in Fig. 5 to demonstrate the viability of a mass transit system in a different city on a different continent.

IV. CONCLUSION

An optimized subway system model is created for any city by combining a K-means clustering and polynomial regression algorithm to be able to determine transit corridor placement throughout a city. Polynomial regression is conducted on each cluster, resulting in distinct transit lines. These algorithms are used on a set of scored grid points generated using a grid search function. A heuristic is used to connect the transit lines together. Stations are placed in the order of highest score to lowest score while maintaining a minimum distance from one another. The results show that the model for this experiment outperforms the current TTC system for $N_s \leq 15$.

For future improvements, a larger data set can be used to ensure that all high-traffic areas are considered. This also ensures that smaller cities, such as Kingston, have more accurate lines. An improvement to improve the defined bounds of a city can also be considered. This ensures that lines and stations strictly appear over areas where transit can be built. The bounds should also take into account locations of water.

REFERENCES

[1] L. S. News Science, "Slime Mold Grows Network Just Like Tokyo Rail System," Wired, Jan. 22, 2010. https://www.wired.com/2010/01/slime-mold-grows-network-just-like-tokyo-rail-system/
[2] A. Fabrikant, "Predicting Bus Delays with Machine Learning," Google Research, Jun. 27, 2019. https://ai.googleblog.com/2019/06/predicting-bus-delays-with-machine.html
[3] City of Eau Claire, "Transit Centre Site Selection Study," City of Eau Claire, Eau Claire, Wisconsin, United States, May 2016. Available: https://wisconsindot.gov/Documents/doing-bus/local-gov/astnce-pgms/transit/ec-site.pdf
[4] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, no. 85, pp. 2825–2830, 2011, Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
[5] A. Que, "Mathematics of Polynomial Regression," polynomialregression.drque.net, 2021. http://polynomialregression.drque.net/math.html

Quantum Convolutional Neural Networks

Chris Kingsland, *Queen's University*, 19cek2@queensu.ca
Abdellah Ghassel, *Queen's University*, abdellah.ghassel@queensu.ca
Taylor Balsky, *Queen's University*, taylor.balsky@queensu.ca
Elad Dallal, *Queen's University*, 20eyd@queensu.ca
Jacob Ewaniuk, *Queen's University*, jacob.ewaniuk@queensu.ca

***Abstract*—In response to the current era of noisy intermediate-scale quantum computing, research into near-term quantum machine learning algorithms has expanded rapidly. Here, we follow this movement by designing two quantum convolutional neural networks (QCNNs) in increasing complexity, and apply them to a multi-class image classification task to fill a void in the field's understanding of current non-hybrid QCNN capabilities. The simpler QCNN architecture was suitable to run on openly-available quantum hardware, and achieved 93% accuracy in binary classification of MNIST handwritten digits. This improved to 98% for a more complex, branching QCNN architecture that provides a look at near-term improvements. Though the accuracies decreased sharply as more classes were added to the classification task, both QCNN architectures were able to learn the task to varying degrees of success, always outperforming random guessing. Altogether, these results promote the use of QCNNs for simple machine learning tasks, and emphasize the need for progression in physical quantum hardware to fully unveil the potential of quantum machine learning.**

I. Introduction

In 2019, Google demonstrated quantum advantage using a programmable quantum processor of 53 superconducting qubits [1]. More recently, IBM revealed a 127-qubit quantum computing chip [2], cementing the notion that the era of noisy intermediate-scale quantum (NISQ) computers is among us. Here, we perform an analysis that contributes to the understanding of quantum machine learning with NISQ technology, following a shift in focus from developing quantum computers to using those available right now.

Many quantum machine learning algorithms feature intrinsic noise robustness and are thus promising candidates for NISQ computers [3]. These algorithms commonly draw inspiration from their classical counterparts, yet are mapped to variational quantum circuits. Classical convolutional neural networks (CNNs) have become the most widely-used deep learning architecture for pattern recognition in recent years, however, they continue to require extensive computational power during training and highly complex models to maintain accuracy [4]. To address these issues, and unveil the potential of quantum feature maps, the quantum convolutional neural network (QCNN) was proposed in Ref. [5]. While this work focused on purely quantum applications, QCNNs can also yield benefits when applied to classical data, particularly in computer vision [6]. Specifically, QCNNs applied to both image processing and recognition tasks have yielded an exponential reduction in gate complexity while obtaining comparable levels of accuracy with many fewer parameters [7].

A. Motivation

With the advent of the NISQ era of quantum computing, alongside the vastly expanding research into quantum machine learning, the development of QCNN architectures and training procedures for relevant near-term applications is an imperative next step. Here, we consider multi-class image classification as an example of mapping conventional machine learning tasks to QCNNs. Previously, both hybrid quantum-classical CNNs [8], [9] and variational quantum classifiers [10], [11] have been developed for such tasks involving more than two classes, however, little research is available for purely quantum CNNs. Separately, branching QCNNs have been proposed to increase the parameter space for training without additional noisy quantum resources (i.e. qubits, gates) [12], however, these networks have only been tested for purely quantum applications. Altogether, there is a gap in our knowledge regarding the efficacy of QCNNs, and their branching counterparts, for multi-class image classification on NISQ hardware. Our goal was to fill this void through the design and analysis of our own QCNN architectures.

B. Related Works

QCNNs, and their branching counterparts, were first proposed in Refs. [5] and [12] respectively, yet neither considered applications from conventional machine learning. In contrast, Refs. [6], [7] consider binary image classification using non-hybrid QCNNs, while Refs. [8], [9] and [10], [11] consider multi-class image classification using hybrid quantum-classical CNNs and variational quantum classifiers respectively.

C. Problem Definition

As quantum computing remains an emerging field, many of its applications are still being researched, having yet to be validated through rigorous testing and implementation. Consequentially, the full extent and true capabilities of QCNNs remain unknown. The transition into the quantum world has been propelled by exploration of hybrid models, combining well-known classical computing algorithms with cutting-edge quantum technology. Though these systems have been well-developed and analyzed, complex projects executed on a purely quantum basis are still lacking in availability. As a result, the true capabilities of advanced algorithms, such as branching QCNNs for conventional machine learning tasks, remain unknown. Furthermore, previous QCNN investigations have only considered binary classification. Evidently, there remain many unresolved questions surrounding non-hybrid

QCNN architectures which make it difficult to understand their potential both in the NISQ era and beyond. Here, we will tackle some of these gaps by exploring both non-branching and branching QCNNs for multi-class image classification, using the MNIST handwritten digit dataset as a benchmarking tool.

II. Methodology

Our QCNN was designed initially using Qiskit in Python such that it could be transferred to real quantum hardware once trained. The structures of each quantum layer in the network were first defined, then parameterized such that the QCNN could be trained to classify images. In a quantum architecture, such parameterization is encoded into the rotation angles of quantum gates.

With the defined structure for each of the convolutional, pooling, and fully-connected layers, the circuit is compiled as a single ansatz for the learning task. This ansatz will operate on input classical data which must be encoded into the qubit states. In the initial, non-branching QCNN, Z feature map encoding was applied, where each qubit is initialized to $|0\rangle$ and then transformed using the Pauli-Z gate, which applies a phase flip to the state if the corresponding data element is non-zero, resulting in highly-entangled inputs [13], [14].

Once the QCNN is able to operate on input classical data, it can be trained. In this implementation, we mapped certain measured computational basis states to classifications. For example, when classifying eight different handwritten digits from the MNIST dataset with the measurement of three qubits, state $|0000\rangle$ implies the image is classified as 1, $|0001\rangle$ implies 2, and so on. By comparing the predicted result to that expected, a loss function is computed and optimized using the COBYLA algorithm, which iteratively adjusts the parameters throughout the network to minimize the overall loss [15].

We first applied the non-branching QCNN design to binary classification, then expanded this to multi-class, with the results summarized in Sec. III. Given that our QCNN implementation was designed with the NISQ era in mind, we were able to send our trained binary classifier to real quantum hardware provided by IBM. While concrete results were not obtained due to limited availability, it is significant that we were able to successfully run our model on a quantum computer, providing a path to understand, in the near future, the practical benefits and limitations of QCNNs in the NISQ era. In the following subsections, we outline in further detail the chosen network architecture, the methods applied to encode MNIST images to qubit states, and the adaptations made to construct a more complex, branching QCNN for improved classification accuracies.

A. Network Architecture

The QCNN architecture is designed in analogy with classical CNNs, yet leverage the potential benefits provided by quantum computing [5]. Classical filters are replaced with quantum gates that enact unitary transformations on qubits. With these operations, the network is able to leverage parallelism from quantum superposition, and strong correlations from entanglement, when learning to classify images.

The main idea of a classical convolutional layer is to apply parameterized sweeping filters that can be applied to the image to create a feature map [4]. Following the design of Ref. [16], our quantum convolutional layer is built from two-qubit parameterized unit cells (outlined in turquoise in Fig. 1) that are applied to each pair of neighbouring qubits.

A classical pooling layer uses the information found in the feature map which was created by the convolutional layer and condenses the dimensions of the image based on the results [4]. Our quantum pooling layer ansatz was inspired from that of Ref. [17], where parameterized quantum gates are applied to help prepare the network to remove half of its qubits, the source qubits, from the processing scheme. However, as outlined in orange in Fig. 1, we simplified the architecture to be more compatible with NISQ hardware, including just two parameterized rotations and a controlled-NOT gate that prepare the source qubit to be removed, and subsequently pass the information to the sink qubit (i.e. that which remains).

The fully-connected layer of the QCNN was based upon the hardware efficient ansatz of Ref. [18]. This circuit is designed to optimize performance while minimizing the number of gates and operations performed on the qubits. This layer operates on the three sink qubits output from the pooling layer, with a structure consisting of single-qubit rotations followed by controlled-NOT gates. Two instances of this arrangement constitute the entire fully-connected layer. The operations performed on the qubits through this layer are meant to further alter and connect their states, preparing them for measurement, at which point the input handwritten digit is classified.

B. Data Encoding & Multi-Class Classification

Quantum embedding involves taking classical data and converting it to quantum states in a Hilbert space [19]. Two common approaches include basis and amplitude encoding. Basis encoding enables us to take classical data and encode it in a specific quantum state via single-qubit rotations. The Z feature map encoding applied in the non-branching QCNN is a more advanced kind of basis encoding, yet both encode just one classical data point per qubit. In contrast, amplitude encoding instills classical data on the amplitudes of each quantum state in the computational basis of dimension 2^n, where n is the number of qubits. The input state can thus be a superposition of the computational basis states, allowing for one classical data point per state, rather than qubit. While this is theoretically much more efficient, it may be difficult to apply this encoding in general in the NISQ era [19].

Autoencoders are a class of unsupervised networks consisting of both an encoder and a decoder. The encoder consists of a series of dense and convolutional layers that compress the image into a compact form, losing as little information as possible. In our non-branching QCNN implementation, 28×28 MNIST images were reduced to 6×1 vectors to encode in the 6 available qubits, as shown in Fig. 1. The

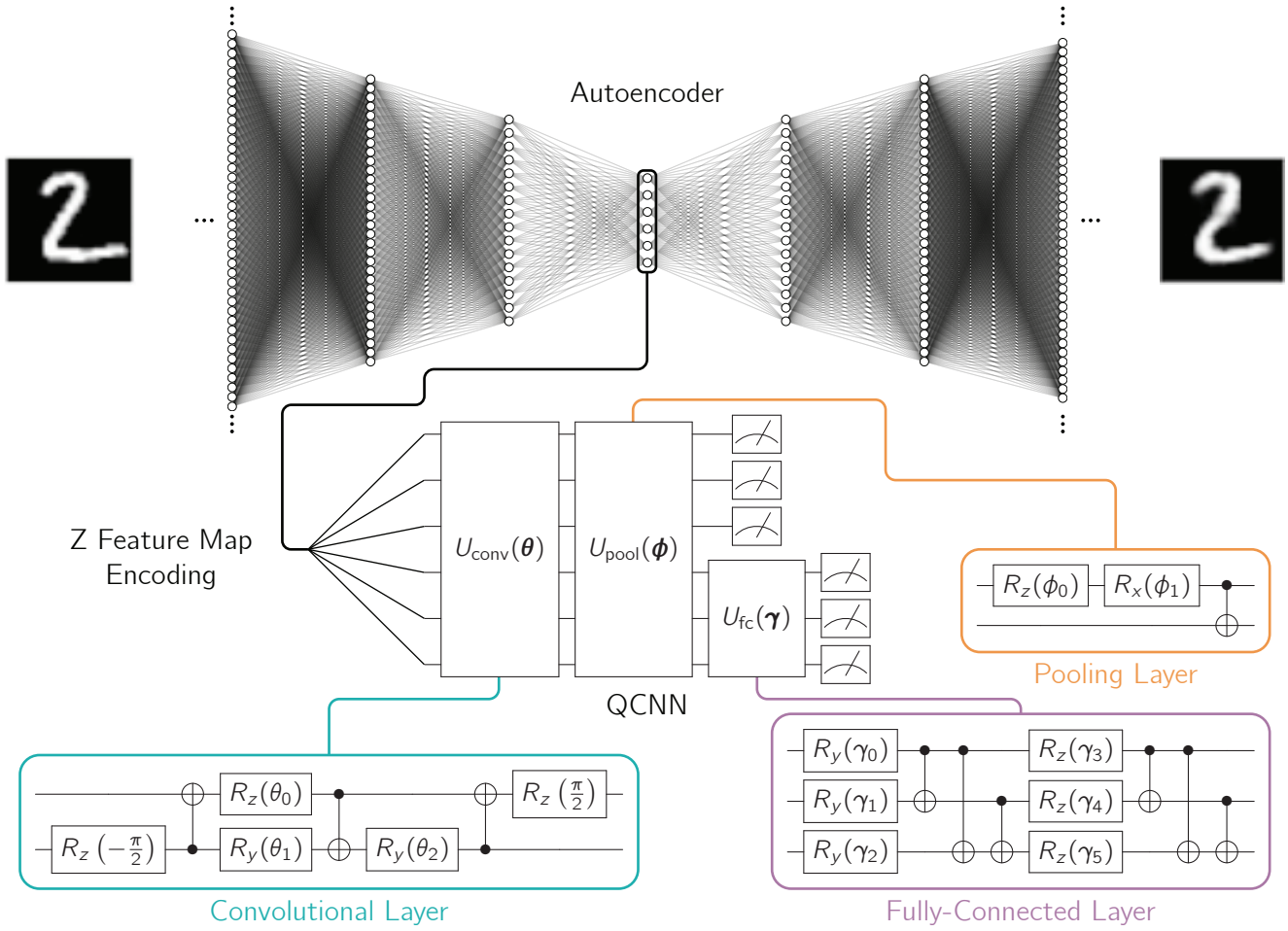

Fig. 1. Network architecture of the non-branching QCNN. An autoencoder compresses MNIST handwritten digits from 28×28 to 6×1, as necessary for Z feature map encoding on 6 qubits. Once the inputs are encoded, the network applies a convolutional layer. This layer features a series of two-qubit unit cells, designed as outlined in turquoise, applied to each neighbouring pair of qubits. A pooling layer is then applied between the source and sink qubits, where for each source and sink, the two qubit unit cell outlined in orange is applied. The source qubits no longer influence the processing and are thus simply measured. In contrast, the fully-connected layer, with corresponding circuit ansatz outlined in purple, is applied to the sink qubits. Classification results are then interpreted from the measurement of the sink qubits.

decoder converts the compressed vector back into the original size. With our autoencoder, a minimal loss of 0.176 was achieved, using a series of dense layers, dropout to prevent overfitting, and LeakyReLu as the activation layer to introduce nonlinearity. It was optimized using a variation of the Adam optimizer, nAdam, which leads to faster and more stable convergence during training [20]. Further improvements can be made by increasing the number of epochs during training and implementing a variational autoencoder to generate new data points and explore the dataset more effectively.

Multi-class classification is much more difficult to learn than binary, however, it is also applicable to wider range of problems. To explore this task, we chose specific types of digits from the MNIST dataset during training. Specifically, for binary classification, only images labelled as 1 and 2 were selected, while for ternary classification, those labelled as 1, 2, and 3 were selected, and so on up to eight total classes.

C. Branching Capabilities

The idea behind branching QCNNs is to increase the parameter space available during training without requiring additional noisy quantum resources that may be unavailable in the NISQ era [12] Here, we apply this idea by designing a more advanced version of the QCNN shown in Fig. 1 that looks toward the near future of NISQ machine learning. As displayed in Fig. 2, this branching QCNN operates on 12 qubits, features two consecutive pairs of convolutional and pooling layers, and applies amplitude encoding, where inputs are programmed into the probability amplitudes of each computational basis state. With 12 qubits, the computational basis has a dimension of $2^{12} = 4096$, thus allowing the input MNIST images to be interpolated from 28×28 to 64×64, rather than compressed to 6×1. The pooling layer design was iterated to include gates that are controlled by the state of the source qubit, thus mimicking the process of measuring

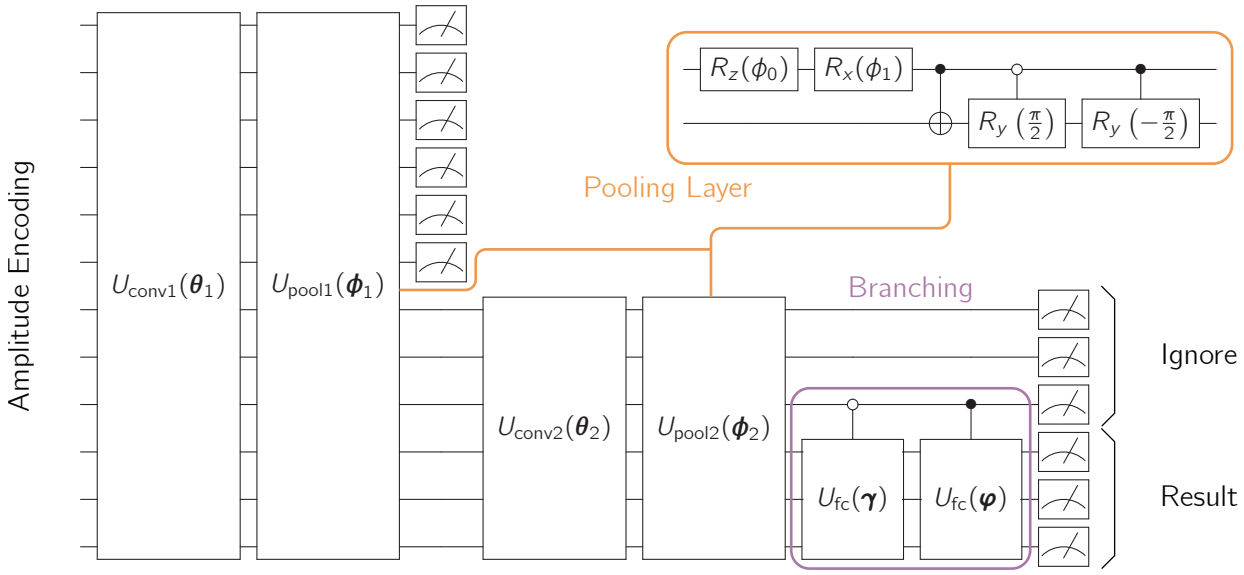

Fig. 2. Network architecture for the branching QCNN. Via the use of amplitude encoding, input MNIST images can be encoded without an autoencoder. Convolutional layers are applied using the same design principle as in Fig. 1. Pooling layers feature the same arrangement between source and sink qubits (i.e. top to bottom), yet each unit cell features additional controlled-gates as outlined in orange. These layers reduce the number of qubits from 12 to 6, then 6 to 3, in preparation for the application of the fully-connected layer, which again follows the same circuit ansatz as Fig. 1. Here, however, different parameters are passed to the fully-connected layer depending on the state of the final source qubit. Regardless, the results are still interpreted from the measurement of the final three sink qubits.

the source and directly projecting its result to the sink.

Branching is applied at the fully-connected layer. Each version of the fully-connected layer shown Fig. 2 follows the same circuit ansatz, yet has its own set of parameters which may be trained. Therefore, in hardware, the two sets of gates outlined in purple can be realized by a single physical fully-connected layer. It is the measurement of the final source qubit that determines which set of parameters ($\boldsymbol{\gamma}$ or $\boldsymbol{\varphi}$) to apply. Given that the circuit ansatz for the fully-connected layer features 6 parameters, the addition of this branching capability alone provides the QCNN with 6 more parameters to train without any further physical resources.

III. Results

We trained both our non-branching and branching QCNN designs in simulation to perform multi-class image classification of MNIST handwritten digits on varying numbers of classes. The maximum recorded classification accuracies, on training sets of 100 images, are displayed in Fig. 3, alongside the accuracy that would be expected by randomly guessing the class of a given input. It is evident that both the non-branching and branching QCNNs were able to achieve higher classification accuracies than random guessing and thus both learned how to classify the data.

For binary classification, the non-branching QCNN achieved 93% accuracy, which improved to 98% for the branching architecture. However, the accuracies sharply decreased for both architectures as the number of classes increased. Specifically, when classifying digits 1-8, the non-branching and branching QCNNs achieved 20% and 26% accuracies respectively. Therefore, while the amplitude encoding, increased numbers of qubits and layers, and branching capabilities yielded improvements in accuracy as expected, these design changes were not sufficient to fully enable high-accuracy multi-class

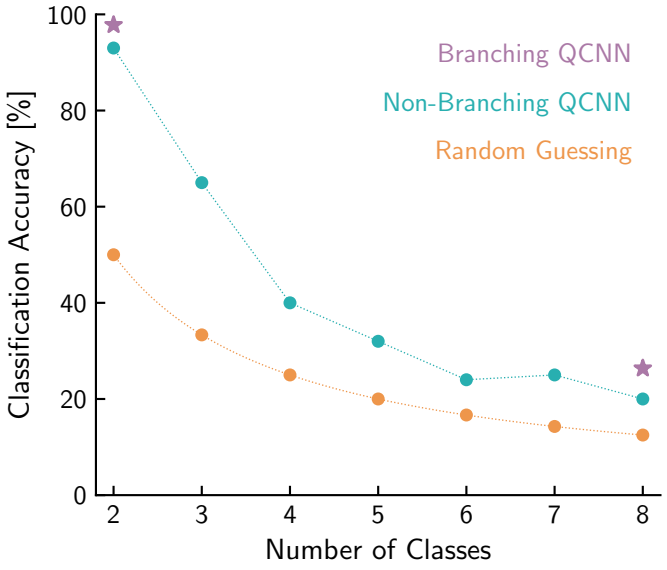

Fig. 3. Accuracies in classifying MNIST handwritten digits with a non-branching QCNN (turquoise stars), its branching couterpart (purple circles), and by randomly guessing (orange circles), for increasing numbers of classes (i.e. types of digits 1-8). Each of the QCNN results show the maximum recorded accuracy on the training dataset, where each contained 100 images. The dotted lines serve only as a visual aid.

image classification. A significant increase in quantum resources is thus likely required to compete with both conventional machine learning approaches and hybrid architectures.

IV. Conclusion

In conclusion, we designed and trained two QCNN architectures in increasing complexity, and benchmarked their performance for multi-class image classification using the MNIST handwritten digit dataset. Our results work to fill a void in the understanding of using purely quantum CNNs for conventional machine learning tasks. Specifically, we discovered that QCNNs readily available in the NISQ era of quantum computing can achieve high-accuracy ($\geq 93\%$) binary image classification, and were even able to demonstrate this availability by running our trained architecture on physical quantum hardware. These results emphasize the promise of quantum machine learning, even in the NISQ era, and promote the need for further research into improved algorithms as well as encoding and training procedures.

When expanding to higher numbers of classes, however, we found that QCNNs require more than just amplitude encoding and branching capabilities to improve their performance. Additional quantum resources are required to achieve high-accuracy classification. Even with 12 qubits, and 5 total layers, as in our implementation of the branching QCNN, simulating these networks on classical computers is a daunting task. Therefore, it is up to the field to continue improving and scaling actual quantum computers. It is only with the development of large-scale fault-tolerant quantum computing that the true potential of QCNNs, and quantum machine learning in general, will be unleashed.

References

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, October 2019.

[2] P. Ball, "First quantum computer to pack 100 qubits enters crowded race," *Nature*, vol. 599, November 2021.

[3] P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Phys. Rev. A*, vol. 98, p. 012324, July 2018.

[4] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, May 2018.

[5] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, pp. 1273–1278, August 2019.

[6] V. Rajesh, U. P. Naik, and Mohana, "Quantum convolutional neural networks (qcnn) using deep learning for computer vision applications," in *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pp. 728–734, October 2021.

[7] S. Wei, Y. Chen, Z. Zhou, and G. Long, "A quantum convolutional neural network on nisq devices," *AAPPS Bulletin*, vol. 32, p. 2, January 2022.

[8] D. Bokhan, A. S. Mastiukova, A. S. Boev, D. N. Trubnikov, and A. K. Fedorov, "Multiclass classification using quantum convolutional neural networks with hybrid quantum-classical learning," *Frontiers in Physics*, vol. 10, November 2022.

[9] A. Chalumuri, R. Kune, and B. S. Manoj, "A hybrid classical-quantum approach for multi-class classification," *Quantum Information Processing*, vol. 20, p. 119, March 2021.

[10] M. Lazzarin, D. E. Galli, and E. Prati, "Multi-class quantum classifiers with tensor network circuits for quantum phase recognition," *Physics Letters A*, vol. 434, p. 128056, May 2022.

[11] Y. Du, Y. Yang, D. Tao, and M.-H. Hsieh, "Demystify problem-dependent power of quantum neural networks on multi-class classification." arXiv, December 2022.

[12] I. MacCormack, C. Delaney, A. Galda, N. Aggarwal, and P. Narang, "Branching quantum convolutional neural networks," *Phys. Rev. Res.*, vol. 4, p. 013117, February 2022.

[13] J. Qin, "Review of ansatz designing techniques for variational quantum algorithms." arXiv, December 2022.

[14] M. M. Hossain, M. S. Ali, R. A. Swarna, M. M. Hasan, N. Habib, M. W. Rahman, M. M. Azad, and M. M. Rahman, "Analyzing the effect of feature mapping techniques along with the circuit depth in quantum supervised learning by utilizing quantum support vector machine," in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, pp. 1–5, December 2021.

[15] Y. Huang, H. Lei, and X. Li, "An empirical study of optimizers for quantum machine learning," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, pp. 1560–1566, December 2020.

[16] F. Vatan and C. Williams, "Optimal quantum circuits for general two-qubit gates," *Phys. Rev. A*, vol. 69, p. 032315, March 2004.

[17] T. Hur, L. Kim, and D. K. Park, "Quantum convolutional neural network for classical data classification." arXiv, February 2022.

[18] O. Kyriienko, A. E. Paine, and V. E. Elfving, "Solving nonlinear differential equations with differentiable quantum circuits," *Phys. Rev. A*, vol. 103, p. 052416, May 2021.

[19] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Phys. Rev. Lett.*, vol. 122, p. 040504, February 2019.

[20] T. Dozat, "Incorporating nesterov momentum into adam." OpenReview, February 2016.

Monte Carlo Tree Search and Reinforcement Learning for a Four Player, Simultaneous Move Game

Finn Archinuk
University of Victoria
finn.archinuk@gmail.com

Dana Bell
University of Victoria
danabell841@gmail.com

Leo McKee-Reid
University of Victoria
leo.tzu.mr@gmail.com

Eric Showers
University of Victoria
ejshowers@gmail.com

Nathan Woloshyn
University of Victoria
nathanwoloshyn@gmail.com

***Abstract*—This work introduces a variant of Monte Carlo Tree Search (MCTS) for the game *BattleSnake*, a 4 player adversarial grid-based game with simultaneous turns. The solution we propose solves the problem of node selection in MCTS by having each player select a subset of nodes to explore instead of deterministically selecting a node, which is how traditional MCTS works. Our implementation allows an MCTS-based Reinforcement Learning model to iteratively improve against previous generations and benchmark models.**

I. Introduction

The study of game-playing AI is as old as computer science itself. Turing, Shannon and Von Neumann were all fascinated by the prospect of algorithms which could play games such as chess at a human (or superhuman) level. For many years, chess was a grand challenge for AI researchers, which culminated in the historic victory of Deep Blue over Gary Kasparov. But this victory can be attributed to sophisticated search algorithms and advanced hardware, not machine learning. Systems like Deep Blue are highly tuned to their domain, and rely heavily on the domain expertise of their programmers.

The ambition of much of modern AI research is to instead create general, blank slate systems that can learn a strong policy from first principles. Recently there have been impressive results from the application of Reinforcement Learning (RL) from self-play using tree search as a guide [Sil+17b]. Games such as Chess, Go, and Shogi ([Sil+17a] have superhuman quality agents. An important feature is that these games are between two opponents making sequential moves, so the traversal into new states is clearly defined. Similarly, for single player games the agent acts at each layer of the tree instead of alternating with the opponent.

Developing algorithms to solve these single player tasks can lead to real-world applications; a high profile example is AlphaTensor, where the researchers found an agent that generates a series of operations to more efficiently multiply matrices [Faw+22].

In this paper we develop a variant of tree search to train a Reinforcement Learning (RL) agent in the popular programming competition *BattleSnake*. We implement this algorithm and demonstrate the iterative improvements of the learned policy.

A. BattleSnake

BattleSnake is a turn based, simultaneous action, strategy game/programming competition inspired by the classic *Snake* arcade game. The *BattleSnake* environment is designed as a starting point for programming projects. Each of four players controls a "snake" with code that selects their move in one of four cardinal directions. The 11x11 board state is updated synchronously. During play a snake can consume food to extend its body length by one and refill a slowly decreasing health bar. A snake dies when it collides with a wall, has not eaten for 100 turns, or collides with a body segment of itself or another snake. If two or more snakes have a head-to-head collision, the largest snake survives.

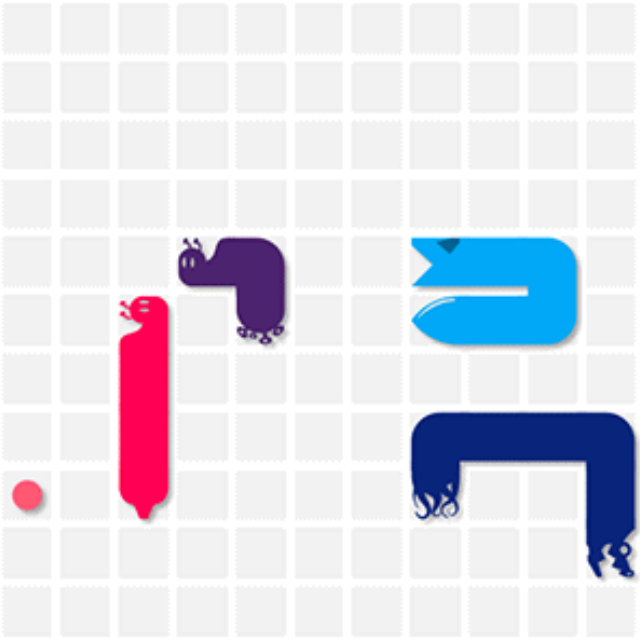

Fig. 1. A frame of BattleSnake with four agents on an 11x11 board.

In an official competition setting a player receives a data structure containing the current board state from the server hosting the match and has 500ms to respond with their selected action (failing to respond in time results in your snake

repeating it's previous move). The time limit constrains the amount of look ahead that can be done using tree search algorithms, such as minimax. Previous algorithms that have proven to be successful are flood-fill (an algorithm that finds connected positions) and A* (an algorithm that efficiently moves to a target).

The theoretical complexity of *BattleSnake* is high in terms of branching factor, which is the number of possible state transitions from any given state. However, this can be significantly pruned in practice. If all agents are alive, each one has four possible moves. Additionally, one item of food may or may not spawn randomly on any empty tile each turn. This gives us about $4^4 * (121 + 1) = 31232$ potential transitions. But we can ignore most of them by assuming that snakes will avoid moves that make them lose instantly (such as hitting their own neck or a wall), and that food spawns are irrelevant for search purposes. This reduces the search space to a much more manageable $3^4 = 81$ at most, and often even less (for example, when some agents are dead or have only one viable move). It is worth noting however that even this curtailed space is often still significantly higher than that of chess (≈ 35), and that the time control is much stricter than is usually the case in chess.

These constraints make this an interesting problem for RL; multiple agents, synchronous moves, and a high branching factor make exhaustive tree search computationally impossible. The complexity is further increased by the stochastic addition of food at random locations at random intervals, and some tree search methods are unable to handle randomness.

B. Problem Definition

The implementation of MCTS from *AlphaZero* is not immediately capable of handling *BattleSnake* due to multiple agents jointly defining a state transition. Variants of MCTS for multiple agents have been proposed before, and will be further outlined in section I-C. Our proposed solution recognizes that the quality of a move of a snake will depend on the moves of other snakes, and that each snake selects a reduced set of states that benefits them the most. During a real game (i.e., not for training), once each snake has made a selection, the intersection of the sets of states is reduced to one, which is the transition to the new state. However, when exploring the game tree with MCTS for training purposes, choosing a set of moves to transition to the next game state is far more complicated.

Creating an expert agent in *BattleSnake* is an important problem to solve, as many problems in the physical world can be thought of as multi-agent, simultaneous action games (either zero sum like *BattleSnake*, or with elements of co-operation), such as traffic on a road or mobile robots in a warehouse.

The goal of MCTS-based RL is to train a neural network (NN) to approximate the resulting MCTS tree. This paper outlines and implements a variation of MCTS that is applicable for multiple agents with simultaneous moves that is appropriate for *BattleSnake*.

C. Related Works

With the recent renaissance in deep learning, there have been many new ways of applying RL to problems such as computer Go, which have branching factors that limit the effectiveness of traditional tree search algorithms, such as minimax. Deep Blue's defeat of Kasparov in 1997 showed that sufficient amounts of compute applied in clever ways can surpass the strategic planning abilities of the strongest human players, but it used inflexible domain specific systems. *AlphaZero*, however, uses a general framework, using a deep convolutional network to approximate the move probabilities of a Monte Carlo search at each position.

Tak et al [TLW14] proposed Simultaneous Move Monte Carlo Tree Search (SM-MCTS) for a variety of two player games. They evaluate multiple formulations of the problem including as *Decoupled Upper Confidence Bounds for Trees*, *Exp3*, *Regret Matching*, and *Sequential Upper Confidence Bounds for Trees*. Their goals differ from what we are proposing in that we intend to approximate the resulting tree with a NN for iterative improvements.

Lanctot et al [Lan+13] applied MCTS to *Tron*, a game that resembles *BattleSnake* in some aspects. In this game, players move simultaneously and win by surviving longer than their single opponent while avoiding the walls they leave behind on the board. They used two implementations of MCTS: one that models the game sequentially during tree traversal but plays Monte Carlo simulations simultaneously, and another that uses SM-MCTS. They experimented with different algorithms for growing the tree for SM-MCTS, such as DUCT, Exp3 and Regret Matching. Decoupled UCT (DUCT) selects moves sequentially but hides the other agent's choice until both agents have chosen, simulating simultaneous play. Exp3 samples moves from a probability distribution based on their expected reward. Regret Matching generates a policy by comparing the regret values of moves. Our work is most similar to Exp3, but we differ in using a learned policy (for both tree construction and simulations) with a neural network.

There was a previous effort by a UVic team to apply the *AlphaZero* architecture to *BattleSnake*, but they eschewed the use of MCTS in favor of a pure value-based RL approach, citing issues with mustering the necessary compute for an MCTS based training pipeline [Sid+20]. In the domain of *BattleSnake*, current SOTA methods are heavily reliant on alpha-beta search methods with clever heuristics to maximize the amount of look-ahead that can be done in the 500ms response window.

II. Methodology

A. Coordinate System

The data structure provided at each turn comes in the form of a dictionary. We have represented the information in the dictionary as a series of layers, with 3 layers per snake (head location, body segment locations, and health) and a food layer. This results in an (11,11,13) array for a 4-player game on the standard (11,11) game board.

The number of board states can be reduced by a factor of 4 by converting this array into a relative coordinate system. This is inspired by [Sid+20] and is done by padding the board state to a $2n-1$ square grid (where n is the width of the board), translating the head of the selected snake to the center, and rotating so that the snake faces up. Padding the board increases the dimensionality of the board, but the complexity of the task is reduced since "nearby" obstacles are always towards the center of the board. Rotating the board reduces a snake's action space from the 4 cardinal directions into 3 relative directions (L=left, F=forward, R=right) since a downwards move will always collide with it's neck. To ensure the snake doesn't run off the board, we also add a channel indicating which locations are walls, so the final board dimensionality is (21,21,14).

B. Traditional MCTS

Monte Carlo Tree Search was introduced in [Abr86] to solve the problem of creating trees for an intractable problem. The challenge with other tree methods are that the number of nodes of the tree may expand beyond what memory can handle. The MCTS method iteratively builds a tree using simulations from leaves to estimate the quality of the state. While the MCTS method does not create a "true" tree, it will converge to the optimal tree as more samples are taken. Furthermore, by selecting the promising moves instead of an exhaustive search of the tree, the algorithm indirectly incorporates pruning.

Algorithm 1 Traditional MCTS

```
while within time limit do
    currentNode ← rootNode
    while currentNode ∈ searchTree do
        lastNode ← currentNode
        currentNode ← SELECT(currentNode)
    end while
    lastNode ← EXPAND(lastNode)
    Reward ← SIMULATION(lastNode)
    while currentNode ∈ searchTree do
        currentNode.BACKPROPAGATE(Reward)
        currenNode.visitCount ← currentNode.visitCount+1
        currentNode ← currentNode.parent
    end while
end while
```

MCTS can be broken down into 4 phases: selection, expansion, simulation, and backpropagation. Given the current tree, a node is *selected* that balances exploitation of previous rewards and exploration of interesting states. Selection is continued until the algorithm finds a leaf. The leaf is *expanded* to include all possible children. One of those children has a *simulation* run to approximate the quality of that state. Finally, the quality of the state is sent back up the tree through *backpropagation*. The MCTS algorithm is more thoroughly outlined in [Bro+12] and [Świ+22].

The primary challenge of implementing MCTS for *BattleSnake* is the *selection* phase, which we discuss in greater detail in the following section. The specific challenge is in selecting a node that greedily satisfies all 4 agents while also exploring the game tree for very good moves which are difficult to find.

C. Multiagent Simultaneous MCTS

With Multi-Agent MCTS [ZY19] and Simultaneous Move MCTS [Lan+13] having both been developed, we propose a combination of the two in order to effectively learn simultaneous multi agent environments: Multi-Agent Simultaneous MCTS (MAS-MCTS). The algorithm supports any number of agents simultaneously selecting moves.

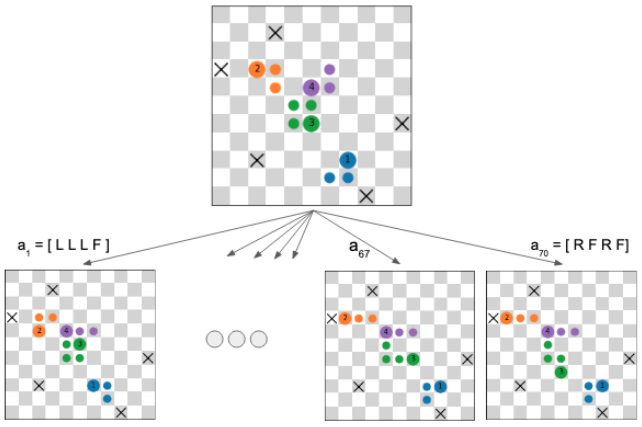

Fig. 2. Transitions from a parent to a child are in the form of a 4 element action.

In the case of *BattleSnake*, we support 4 agents selecting one of up to 3 moves each turn, which collectively define an action and state transition. Therefore our maximum non-trivial branching factor from any state is: $\#\ moves^{\#\ agents} = 3^4 = 81$. Agents select a move by considering what state transitions could possibly result from that move, which is the set of all possible actions where that agent chose that move. For example, for an agent to consider moving left they would consider the set of state transitions where they move left, and every other agent can make any of their valid moves. In the simple case of two agents, this would be the set $\{(L,L),(L,F),(L,R)\}$, where our agent moves left and the other agent may move left, forward or right. Figure 2 shows possible state transitions for *BattleSnake*. The arrows indicate state transitions as a combination of moves of the snakes. If a snake chooses to move left, there are 27 possible resulting states due to the moves of the opponents.

The selection of which child node to visit is collectively decided by the agents using a combination of *exploitation*—how likely an agent is to win from that node—and *exploration*—a measure of how novel the node is. The exploitation component is defined in equation 1 as the total reward collected from travelling to that node (W_a) divided by the number of visits to that node (N_a), repeated for each agent. This results in a reward summarization of the set of moves for each snake. The reward summarization is mapped to the 81 child nodes by

iterating over all permutations. This value is then normalized to sum to one. Note that in the following equations, a is an action, where an action is defined as a combination of all the agents moves. Actions therefore represent the edge from a node to a child node in the game tree.

$$Q_a = \frac{W_a}{N_a} \tag{1}$$

The exploration component, U, is defined in equation 2. The exploration coefficient (c) multiplies the NN probability of selecting a given action, $P(s,a)$, which is the product of the probability that the NN assigns to that move for each snake (equation 3). We therefore call the NN, $f_\theta(s)$, at every state when calculating the exploration component. For example, given the NN probability of each move for each snake at state s in Table I, if some action a = [L, L, R, F], then $P(s,a)$ = $m_{1,1}$ * $m_{2,1}$ * $m_{3,3}$ * $m_{4,2}$

TABLE I
NN OUTPUTS USED FOR CALCULATING $P(s,a)$

Alive Snakes	Move Left	Move Forward	Move Right
$f_\theta(s)$ for Snake1	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$
$f_\theta(s)$ for Snake2	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$
$f_\theta(s)$ for Snake3	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$
$f_\theta(s)$ for Snake4	$m_{4,1}$	$m_{4,2}$	$m_{4,3}$

In this table, $m_{i,j}$ is the probability that the NN assigns to the i^{th} snake to move in the j^{th} direction

The remainder of equation 2 biases exploration of under-visited nodes by dividing the number of visits of the children ($\sum_b^{81} N(s,b)$) by the number of all visits from that state ($1+N(s,a)$).

$$U_a = cP(s,a)\frac{\sqrt{\sum_b^{81} N(s,b)}}{1+N(s,a)} \tag{2}$$

$$P(s,a) = \prod_i Pr(f_\theta(s) = a_i) \tag{3}$$

$\forall i$ such that $snake_i$ is currently alive

The selection of the node to visit is done by finding the maximum Action Value $V_A = Q + U$. As with traditional MCTS, if the selected node has child nodes, this process is repeated. If the selected node does not yet have children, the node is expanded, and the Action Value of the child nodes are calculated.

At the end of each MCTS loop for a given state, the selected action to advance one turn in the real game is calculated by taking a probability distribution of π 4. π is a 4x3 array, where each element is proportional to how often a given snake made a certain move during MCTS. This selection method makes sense, since the number of visits to a node during MCTS, N_a, is determined by both the exploitation and exploration components.

$$\pi_{i,j} = \sum_{a:a_i=j} \frac{N_a}{\sum_b N_b} \tag{4}$$

The following algorithm 2 describes the repeated process of generating data and training the NN via self-play.

Algorithm 2 MCTS Training Loop

```
Generate initial set of training games using trivial policy
while NN improving do
    while Games played < 200 do                ▷ Generate data
        Initialize new game with 4 identical NN agents
        while Game is not over do
            Use MCTS and the NN to calculate the best next
              move probability for all agents, π
            Save current game state and associated π
            Move all agents 1 turn as a probability
              distribution of π
        end while
    end while
    Train new NN on generated games
    Ensure new NN out-performs previous NN
    NN ← new NN
end while
```

III. RESULTS

A. Initial Training Data

To generate the initial training data, 200 games were played using MCTS simulations at each state to calculate how each snake would move. For the first set of games, we use a trivial agent that was classically coded to avoid immediately moving into obstacles, but had no other planning routine. For this initial set of games, each turn was simulated 2000 times with a maximum depth of 10 steps or until only one snake remains, whichever comes first. The intention here is to provide high quality training data for the first NN. These games resulted in approximately 270,000 samples for training.

B. Training

The NN is a simple convolutional neural network (CNN) with two convolutional layers (16 filters, 3x3 kernel, ReLU activation), followed by two fully connected layers with 256 neurons each and ReLU activation. There are then 3 output neurons with a softmax activation. This model was trained for 100 epochs with a batch size of 1024. The optimizer was AdamW [LH17] with a learning rate of 5e-5 and weight decay of 5e-5. The output of the NN is a 3 element probability distribution determined by the MCTS summary, so we selected Kullback-Leibler Divergence (KLD) as the loss function.

C. Training Loop

Once the NN was trained on the initial games, it becomes an approximation of MCTS. The next set of games were trained using the fixed weights of the NN. Each turn of these games were simulated only 1000 times, since the planning of the NN could produce higher quality exploration than a random policy. This training loop is outlined in Algorithm 2.

Training the NN requires converting the board state into a relative coordinate frame and reducing the predicted loss using the MCTS generated tree summarization as the ground truth.

D. Architecture Modification

The most recent NN update includes an adjustment to the architecture and a preprocessing step to the inputs. The architecture has an additional fully connected layer of 256 neurons with a ReLU activation. The health of snakes were reduced by a factor of 100. Finally, the snake bodies have an inherent order to them (from neck down to tail) which the network can access. The previous training data had these body segments clipped to either zero or one, indicating either the absence or presence of a body obstacle, respectively. This alternative architecture and preprocessing was selected due to a significant decrease in validation loss observed during training.

E. Evaluation

Here we evaluate the quality of the learned policy with special attention paid to improvements between iterations. The evaluation metric we use is the percentage of wins against three opponents. Since *BattleSnake* has four agents, our learned policy is expected to win 25% of the time if it is performing at a similar level. This is an imperfect metric since snake behaviours may be non-transitive, with snakes winning more or less frequently depending on the behaviours of their similarly matched opponents. As a first approximation, and to demonstrate a trend in improvement, win percentage is sufficient.

At the time of writing we have three iterations of the NN policy. For each of the three iterations we are testing against two types of opponents: a Trivial snake that is only able to avoid immediately adjacent obstacles, and a Flood Fill snake. The Flood Fill snake works by moving to the best non-obstacle position according to a heuristic which calculates the best-case longest-path possible from each available position.

Each iteration plays 500 times against three of the opponents of the given type. Table II shows the learned policy is able to regularly win against 3 Trivial opponents. This table also shows that although it has a low win percentage against the more complex Flood Fill snake, the win percentage increases with each training iteration.

TABLE II
WIN PERCENTAGE VS 3 OPPONENTS

Policy	vs Trivial	vs Flood
Iter. 1	53.4%	2.8%
Iter. 2	62.4%	4.0%
Iter. 3	76.6%	6.8%

To ensure the snakes are improving relative to their previous generation, we also test all three iterations together, with the fourth spot filled with either the Trivial Snake or the Flood Filling snake. The results of this test are summarized in Table III. These values may not sum to 100% due to draws. Again, 500 games were played to limit the influence of random chance.

TABLE III
WIN PERCENTAGE VS VARIED OPPONENT

Game	Policy	Wins
1	Trivial	4.4%
1	Iter. 1	15.8%
1	Iter. 2	23.2%
1	Iter. 3	54.2%
2	Flood	66.6%
2	Iter. 1	5.2%
2	Iter. 2	8.6%
2	Iter. 3	18.8%

These results indicate that not only is each training iteration improving the model, but the improvements from Iter. 2 to Iter. 3 are greater than from Iter. 1 to Iter. 2. Based on these initial results, it is reasonable to anticipate that the model will continue to improve with additional training.

IV. Conclusion

In this work we have outlined a variant of Monte Carlo Tree Search for multiple adversarial agents in a sychronous environment. We have shown this algorithm can be used for training a neural network as introduced by AlphaZero [Sil+17b] in an iterative process, and that later iterations of the model outperform prior iterations.

V. Future Work

While the MCTS-based RL method we have developed is generating good initial results, this project will undergo a number of changes in the coming weeks to improve performance. In addition to experimenting with different NN architectures, hyperparameter tuning, and increasing our depth of MCTS simulation, we will also be updating our training pipeline, as described in the following algorithm.

Algorithm 3 Improved MCTS Training Loop

```
Generate initial set of training games using trivial policy
while NN improving do
    while Games played < 500 do                ▷ Generate data
        Initialize new game
        while Game is not over do
            Use MCTS and the NN to calculate the best next
              move probability for all agents, π
            Save current game state and associated π
            Move all snakes 1 turn as a probability
              distribution of π
        end while
    end while
    Train 3 new NNs on generated games
    Play 100 4-player evaluation games between the 3 new
      NNs and the previous NN
    NN ← the NN which won the most test games
end while
```

Aside from increasing the number of generated training games for each model iteration, the difference between this algorithm and the currently implemented pipeline (Algorithm

2) is the idea of training multiple NNs each iteration, then selecting the best by running evaluation games. Generating games is the most computationally expensive step in this pipeline, so better utilizing the generated games may decrease the number of training iterations needed to achieve an equivalent skill level.

Finally, we will experiment with different rewards during MCTS simulations. Our current model is only rewarded for survival (with less reward if opponents are also alive). Giving explicit reward for eliminating opponents, finding food, or ending games quickly may significantly affect training. Since *BattleSnake* is an incredibly complex and stochastic game, there may be different playing styles that correlated with different peaks of optimal play, and reward engineering may be the key to unlocking certain styles of play.

References

[Abr86] Bruce Abramson. "Thesis Proposal: The Expected-Outcome Model of Two-Player Games". In: 1986.

[Bro+12] Cameron Browne et al. "A Survey of Monte Carlo Tree Search Methods". In: *IEEE Transactions on Computational Intelligence and AI in Games* 4 (2012), pp. 1–43.

[Lan+13] Marc Lanctot et al. "Monte Carlo Tree Search for Simultaneous Move Games: A Case Study in the Game of Tron". In: 2013.

[TLW14] Mandy J. W. Tak, Marc Lanctot, and Mark H. M. Winands. "Monte Carlo Tree Search variants for simultaneous move games". In: *2014 IEEE Conference on Computational Intelligence and Games* (2014), pp. 1–8.

[LH17] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2017. DOI: 10.48550/ARXIV.1711.05101. URL: https://arxiv.org/abs/1711.05101.

[Sil+17a] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. DOI: 10.48550/ARXIV.1712.01815. URL: https://arxiv.org/abs/1712.01815.

[Sil+17b] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550 (2017), pp. 354–359.

[ZY19] Nicholas Zerbel and Logan Michael Yliniemi. "Multiagent Monte Carlo Tree Search". In: *Adaptive Agents and Multi-Agent Systems*. 2019.

[Sid+20] Ahmed Siddiqui et al. "Multiagent Reinforcement Learning in a Synchronous Strategy Game". In: (2020). URL: https://github.com/Fool-Yang/AlphaSnake-Zero/blob/master/report.pdf.

[Faw+22] Alhussein Fawzi et al. "Discovering faster matrix multiplication algorithms with reinforcement learning". In: *Nature* 610 (2022), pp. 47–53.

[Świ+22] Maciej Świechowski et al. "Monte Carlo Tree Search: a review of recent modifications and applications". In: *Artificial Intelligence Review* 56.3 (2022), pp. 2497–2562. DOI: 10.1007/s10462-022-10228-y.

AGORA: a Language Model for Safe Speech-to-Text Conversion

Victor Cruz
McGill University
victor.cruz@mail.mcgill.ca

Laurence Liang
McGill University
laurence.liang@mail.mcgill.ca

***Abstract*—Large language models are trained on immense data sets. This can let some unpleasant language be available for the end user of the language model. With the rise of the proliferation of close captioning software, it becomes increasingly important to ensure the correct interpretation and safe transcription of speech during different events which are required to be safe for everyone. We use GPT-3 to evaluate and rewrite harmful text sequences into harmless ones using zero-shot learning. Using zero-shot learning, we find that GPT-3 can identify and paraphrase text with harmful language. As a result, we introduce the Automated Generation and Omission Recurrent Architecture (AGORA) as one of the first models of its kind to work in conjunction with the new speech recognition model Whisper to provide speech-to-text transcription with built-in offensive content filtration and rephrasing.**

I. Introduction

Text-to-speech models have been notorious for misinterpreting certain words from people with disabilities [1] during different use cases. Furthermore, it is quite important to maintain a safe environment for cases where children are interacting with a model like this [2]. This can become a problem when a user is trying to convey information to other people but their audio is not working well.

A. Motivation

It is quite important to have reliable speech software to interpret what was said, amid growing use cases of video conferencing and streaming. [3], [4] This can help millions of people learn English and other languages and interact with the internet in a safe and guided way.

B. Related Works

1) Speech Recognition: Speech recognition is the process of transcribing words in an audio recording into text. Recent advances in neural networks have attracted considerable interest in developing end-to-end (E2E) models that involve LSTM or Transformer architectures. [5] In particular, one such state-of-the-art model is Whisper, which was developed by OpenAI, whose performances rival those of human actors when transcribing speech to text [6].

2) Zero-shot Prompting with Large Language Models: Zero-shot prompting is an emerging approach to identifying offensive text, which involves assigning a model to a task that it has never seen before. While zero-shot prompting may force a model to draw conclusions based on limited prior knowledge, zero-shot prompting with large language models can be an effective approach to identifying offensive text. Because offensive text can (i) have implicit meanings, (ii) is not specific to certain keywords and (iii) can be interpreted differently from one group to another, it becomes difficult to develop keyword-based models that perform well amid these observations. [7] As large language models are designed differently than keyword-based models, zero-shot learning, for instance with GPT-3, for toxic language detection yields promising results, and few-shot learning can also increase model performance. [8]

3) Text Detoxification: Large language models such as BART and GPT-3 are able to generate high-quality text that can become nearly indistinguishable from human-generated content. [9], [10] However, it becomes important to also investigate methods to ensure that generated text is detoxified. [11] Past research includes developing autoencoder models to mask and rewrite harmful content or fine-tuning large language models (in the case of InstructGPT). [12], [13]

II. Methodology

A. Overall Architecture

We introduce our speech-to-text model, AGORA (Automated Generation and Omission Recurrent Architecture), which transcribes an audio recording into a non-offensive word statement. (Figure 1) AGORA has three components: (i) the speech-to-text component, (ii) the zero-shot offensive language detection component, and (iii) the paraphrasing component.

When AGORA receives an audio recording, it will first transcribe the speech into a transcript using the (i) speech-to-text component, which attempts to transcribe the exact same words in the audio recording.

Next, the (ii) offensive language detection component will assess whether the current text contains any offensive language. If this (ii) component detects no offensive content, the current text transcription will be the final output. If this (ii) component detects offensive content, it will then send the current text to the (iii) paraphrasing component.

The (iii) paraphrasing component will reword the current text while attempting to keep the text's underlying meaning. The (iii) paraphrasing component will then send the reworded text to the (ii) offensive language detection component. As long as the reworded text contains offensive language (as detected by component (ii)), AGORA will iteratively process the text statement in a recurrent manner until no more offensive content is detected. However, to avoid infinite loops where AGORA is

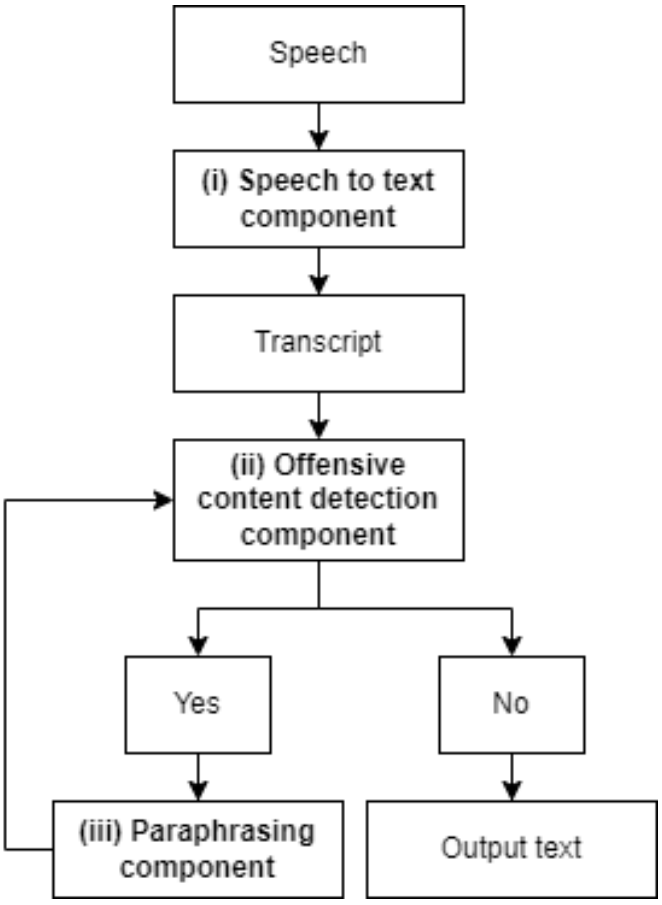

Fig. 1. Overall Architecture of AGORA

unable to adequately reword the subsequent text, AGORA will output the processed text after a maximum of 10 iterations.

B. Speech-to-Text Functionality

The speech-to-text component is built on OpenAI's Whisper, a general-purpose automatic speech recognition model. We selected Whisper due to the size of its training database (680,000 hours of multilingual and multitask samples) and its performance which is close to human-level accuracy. (On the Kincaid46 dataset, Whisper's word error rate (WER) was 8.81%, while the WER for human transcription ranged between 8.14% and 10.5%.) [6]

C. Zero-Shot Offensive Language Detection

To identify whether the current text contains offensive language, the offensive language detection component prompts OpenAI's GPT-3 davinci-003 model with the following instructions:

> Does the following text prompt contain one or several of toxic, severe toxic, obscene, threat, insult or identity hate: {insert text here}

The program then processes GPT-3's output which specifies which of the six attributes are present in the current text.

TABLE I
MULTI-LABEL CLASSIFICATION ON A SUBSET OF THE JIGSAW DATASET

LABEL	PRECISION	RECALL	F1-SCORE
TOXIC	1.00	0.05	0.10
SEVERE TOXIC	0.03	0.67	0.05
OBSCENE	1.00	0.36	0.53
THREAT	0.17	0.50	0.25
INSULT	0.55	0.50	0.52
IDENTITY HATE	1.00	0.33	0.50
MICRO AVG	0.15	0.29	0.20
MACRO AVG	0.62	0.40	0.32
WEIGHTED AVG	0.80	0.29	0.32
SAMPLES AVG	0.04	0.02	0.03

TABLE II
MULTI-LABEL CONFUSION MATRIX ON A SUBSET OF THE JIGSAW DATASET

LABEL	TN	FP	FN	TP
TOXIC	180	0	19	1
SEVERE TOXIC	121	76	1	2
OBSCENE	189	0	7	4
THREAT	193	5	1	1
INSULT	183	5	6	6
IDENTITY HATE	197	0	2	1

AGORA uses the same six attributes (toxic, severe toxic, obscene, threat, insult, identity hate) that were specified in the Jigsaw "Toxic Comment Classification Challenge" in order for AGORA to properly benchmark its model performance. [14] Thus, by design, AGORA can simultaneously identify different labels for a given text (an instance of multi-label classification), as a text can be both obscene and a threat at the same time.

As no prior training or reference prompt is provided to GPT-3, this component's task is zero-shot in nature.

The temperature parameter is set to 0.5.

D. Paraphrasing the Initial Text Prompt

Similar to the offensive language detection component (as highlighted in subsection 3.3), the paraphrasing component prompts OpenAI's GPT-3 davinci-003 with the following instructions:

> Paraphrase the following phrase in a non-offensive way: {insert text here}

This is also a zero-shot task, as AGORA does not provide GPT-3 with any prior training or reference examples.

Similarly, the temperature parameter is set to 0.5.

E. Dataset

We used the "Toxic Comment Classification Challenge" dataset from Jigsaw (on Kaggle) to assess AGORA's ability to detect offensive language. Due to GPT-3 rate limitations, we processed this database's first 200 samples in its training file (159,571 training samples in total). [14] We also sampled 10 audio clips from two movies (*Whiplash* and *The Wolf of*

TABLE III
ATTRIBUTE-BLIND BINARY CLASSIFICATION OF A SUBSET OF THE JIGSAW DATASET (N=200)

ATTRIBUTE	PRECISION	RECALL	F1-SCORE
NON-OFFENSIVE	0.95	0.59	0.72
OFFENSIVE	0.17	0.71	0.27
ACCURACY			0.60
MACRO AVG	0.56	0.65	0.50
WEIGHTED AVG	0.86	0.60	0.68

TABLE IV
CONFUSION MATRIX OF THE ATTRIBUTE-BLIND BINARY CLASSIFICATION (N=200)

LABEL	TN	FP	FN	TP
FREQUENCY	105	74	6	15

TABLE V
AGORA'S END-TO-END PERFORMANCE AS DEFINED BY A HUMAN OBSERVER (N=10)

CRITERIA	ACCURACY
IS THE INITIAL MEANING CONSERVED?	0.6
IS THE FINAL OUTPUT NON-OFFENSIVE?	0.8

Wall Street, where 5 samples were taken from each movie) to assess AGORA's ability to transcribe speech to text, and to subsequently paraphrase the initial text into a non-offensive statement. The audio samples were between 5 to 30 seconds in duration.

F. Evaluation Metrics

We define two respective sets of metrics for the offensive language detection component's accuracy and for the overall model's performance.

1) Metrics for the Zero-Shot Offensive Language Detection: We consider the precision, recall and f1-score to evaluate the multi-label nature of the "Toxic Comment Classification Challenge" database, as multiple attributes can simultaneously exist. (For example, a statement can be both obscene and an insult at the same time.) We also use confusion matrices for each attribute to visualize false positive and false negative occurrences specific to each attribute.

Additionally, we consider these same metrics (precision, recall and f1-score) to evaluate whether AGORA can detect offensive content (any of the six attributes from the Jigsaw dataset), regardless of the specific attribute type. In other words, we evaluate whether AGORA can detect offensive content, even if AGORA fails to correctly classify the type of offensive content in the six aforementioned categories. Consequently, we define this type of task as an attribute-blind binary classification.

2) Metrics for the Overall Architecture: We then evaluate AGORA's end-to-end performance (solely based on the audio input and the final text output) by asking the following two questions

1) Is the initial meaning conserved in the final text output?
2) Is the final text output offensive?

We evaluate AGORA on ten audio samples with offensive content from two movies (as specified in section 3.5).

G. Implementation

AGORA is implemented as a Python class, to promote ease of use across different operating systems and use cases.

III. RESULTS

1) Zero-Shot Offensive Language Detection: At first glance, we notice that AGORA's zero-shot offensive language detection has macro averages of 0.62 for precision, 0.4 for recall, and 0.32 for the f1-score. (Table I) In particular, when analyzing the f1-score, AGORA is best at identifying obscene content (0.53), insults (0.52) and identity hate (0.50). AGORA has the most difficulty with identifying toxic (0.10) and severe toxic (0.05) content, as AGORA has a tendency of missing "toxic"-labeled statements (recall of 0.05) while falsely attributing "severe toxic"-labeled statements (precision of 0.03). (Tables I, II).

However, in an attribute-blind setting where AGORA only needs to identify whether a statement contains offensive content, AGORA's overall accuracy reaches 60%. (Table III) We also note that AGORA tends to label false positives (74) at a much higher rate than false negatives (6). (Table IV) These results suggest that GPT-3 davinci-003 falsely identifies benign text as offensive content (precision of 0.17), though GPT-3 misses offensive content at a much smaller rate (precision of 0.95). The overall accuracy of 60% is comparable to the 56% accuracy for zero-shot learning and the 55% accuracy for one-shot learning in a similar task as reported by [8] when using GPT-3.

2) Speech-to-Text Offensive Content Filtration: As specified in Table V, when evaluated by a human observer, AGORA has a (i) 60% accuracy in conserving the initial meaning of the audio input in the final text output, and reaches an (ii) 80% accuracy in generating a non-offensive final text output. We observe that errors from (i) conserving the initial meaning of the audio file originate in the speech-to-text transcription, paraphrasing the text to the point that it loses its original intent, or a combination of both. (Figure 2) While AGORA filters out explicit profanity such as swear words, AGORA has (ii) difficulty identifying and removing implicit terms, such as sarcasm or passive agressive statements. (Figure 3)

3) Limitations: Due to GPT-3's rate limitations per pricing tier, only the first 200 samples from the Jigsaw dataset were analyzed in this paper. While these results were able to provide a preview of the performance of GPT-3-based architectures, a more comprehensive study of the entire Jigsaw dataset and other additional toxic language datasets would be required to provide a broader and more representative picture in regard to the performance of large language models for paraphrasing toxic text.

Iteration 0: (direct speech to text transcript)
Don't you [expletive] touch it, John! I swear I will [expletive] you!

current iteration: 1
"John, don't touch it! I'm warning you - this is serious!"

current iteration: 2
John, please do not handle that. I am cautioning you - this is important.

current iteration: 3
John, please be careful with that. I'm urging you to take this seriously.

current iteration: 4
John, please exercise caution with that. I strongly suggest you take this seriously.

current iteration: 5
John, please be careful with that. I strongly urge you to take this seriously.

Fig. 2. Wolf of Wall Street excerpt where AGORA has erroneously transcribed "John" instead of "Jordan", and "it" instead of "him". While the final output is non-offensive in nature, the initial meaning is not fully conserved.

Iteration 0: (direct speech to text transcript)
What was that? [expletive] crack! I didn't know they allowed [expletive] into Shaper. Am I to understand that you cannot read Tempo? Can you even [expletive] read music? But I'm gonna do you a [expletive] Acapella group? Play the [expletive] kit!

Iteration 1:
What do you mean? That's strange. I didn't know they allowed anyone who isn't familiar with Shaper into the space. Are you saying that you don't understand Tempo? Can you read music notation? But I'm gonna make an a cappella group for you? Play the drums, please!

Fig. 3. Whiplash excerpt where AGORA has difficulty recognizing sarcastic intent.

As offensive content can be subjective in nature, having multiple human members in a jury would yield more representative results when assessing AGORA's end-to-end performance.

Furthermore, fine-tuning GPT-3 and experimenting with one-shot or few-shot learning are additional approaches that should be investigated.

IV. Conclusion

1) Limitations: Due to GPT-3's rate limitations per pricing tier, only the first 200 samples from the Jigsaw dataset were analyzed in this paper. While these results were able to provide a preview of the performance of GPT-3-based architectures, a more comprehensive study of the entire Jigsaw dataset and other additional toxic language datasets would be required to provide a broader and more representative picture in regard to the performance of large language models for paraphrasing toxic text.

As offensive content can be subjective in nature, having multiple human members in a jury would yield more representative results when assessing AGORA's end-to-end performance.

Furthermore, fine-tuning GPT-3 and experimenting with one-shot or few-shot learning are additional approaches that should be investigated.

2) Final Thoughts: We introduced AGORA, a new speech-to-text model that can filter out offensive content by recursively paraphrasing it. In particular, the comprehensive speech-to-text architecture provides an integrated approach for near-real-time speech-to-text captioning. One-shot and few-shot-learning, expanding the dataset sample size, and fine-tuning language models are future aspects to investigate, in order to provide a more comprehensive overview of jointly using speech-to-text and large language models to filter offensive content.

References

[1] J. T. Garrett, K. W. Heller, L. P. Fowler, P. A. Alberto, L. D. Fredrick, and C. M. O'Rourke, "Using speech recognition software to increase writing fluency for individuals with physical disabilities," *Journal of Special Education Technology*, vol. 26, no. 1, pp. 25–41, 2011. [Online]. Available: https://doi.org/10.1177/016264341102600104

[2] S. Lovato and A. M. Piper, ""siri, is this you?": Understanding young children's interactions with voice input systems," in *Proceedings of the 14th International Conference on Interaction Design and Children*, ser. IDC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 335–338. [Online]. Available: https://doi.org/10.1145/2771839.2771910

[3] "Video conferencing market value worldwide in 2021 and 2026," https://www.statista.com/statistics/1293045/video-conferencing-market-value-worldwide/, accessed: 2022-12-28.

[4] "Video streaming (svod) - worldwide," https://www.statista.com/outlook/dmo/digital-media/video-on-demand/video-streaming-svod/worldwide, accessed: 2022-12-28.

[5] J. Li, "Recent advances in end-to-end automatic speech recognition," 2022.

[6] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://arxiv.org/abs/2212.04356

[7] Y.-S. Wang and Y. Chang, "Toxicity detection with generative prompt-based inference," 2022. [Online]. Available: https://arxiv.org/abs/2205.12390

[8] K.-L. Chiu, A. Collins, and R. Alexander, "Detecting hate speech with gpt-3," 2021. [Online]. Available: https://arxiv.org/abs/2103.12407

[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1910.13461

[10] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[11] C. Xu, Z. He, Z. He, and J. McAuley, "Leashing the inner demons: Self-detoxification for language models," 2022. [Online]. Available: https://arxiv.org/abs/2203.03072

[12] S. Hallinan, A. Liu, Y. Choi, and M. Sap, "Detoxifying text with marco: Controllable revision with experts and anti-experts," 2022. [Online]. Available: https://arxiv.org/abs/2212.10543

[13] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022. [Online]. Available: https://arxiv.org/abs/2203.02155

[14] e. a. cjadams, "Toxic comment classification challenge," 2017. [Online]. Available: https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge

Anomaly Detection for Purefacts Financial Solutions

Sebastian Wakefield
Queen's Univeristy
19sslw@queensu.ca

Connor Rewa
Queen's University
20car8@queensu.ca

***Abstract*—The paper presents a methodology for anomaly detection in financial data, aimed at detecting anomalies in fee calculations in PureFacts Financial Solutions' wealth management solutions. The study employs unsupervised learning approaches, as the data provided was not classified as anomalous/non-anomalous. The authors describe the relevant data parameters used for analysis, including AssignedBPSRate, CurrentGroupValue, StartTier, EndTier, FeeScheduleAssignID, and FeeTypeID. They also discuss the challenges faced in detecting anomalies and present their approach, which involves clustering the data using DBSCAN and then using Davies-Bouldin and Silhouette values to check the optimal epsilon value. The study's results show that the proposed approach successfully detected anomalies in the data, which could save the institution from millions of dollars in fines, regulation costs, and compliance issues.**

I. Introduction

This work focuses on the development of an anomaly detection system for PureFacts Financial Solutions, a company that provides wealth management and asset solutions on an international scale. The accurate calculation of fees charged to clients is crucial for the company to avoid legal or regulatory implications. The paper proposes a system that utilizes unsupervised machine learning approaches such as DBSCAN and K-means to detect anomalies in data records. The paper also discusses algorithmic methods that can be used in conjunction with machine learning models to discover outliers. The results of the proposed approach are promising, and future work will involve further optimization of the system to handle larger datasets. Unfortunately, some of our results cannot be publicized due to an NDA with PureFacts.

A. Motivation

Detecting outliers in data is a large problem that has been studied for many years [1]. Numerous methods have been developed and tested for detecting anomalous points in data sets, as the use of inaccurate data can have critical repercussions [1]. Our client, PureFacts Financial Solutions, provides enterprise wealth management and asset solutions on an international scale. They provide their clients with transformational WealthTech solutions to future-proof their business and accelerate growth by leveraging their expertise in wealth management solutions. Removing anomalous data from PureFacts' data records is imperative to ensure that correct fees are charged to their clients. Incorrect values in PureFacts' records can result in detrimental legal or regulatory implications. Our client needs a reliable system to flag anomalous data for technicians to further review and remove.

B. Related Works

Anomaly detection is a well-studied problem in the field of machine learning, and various methods have been developed to tackle it. One popular approach is to use clustering algorithms to identify data points that are significantly different from others. For example, the k-means algorithm has been used to detect anomalies in time series data [2], while density-based clustering algorithms such as DBSCAN have been used for anomaly detection in network traffic data [3].

Other methods for anomaly detection include using decision trees [4], support vector machines (SVMs) [5], and neural networks [6]. These methods can be effective, but they can also be computationally expensive and difficult to interpret.

One challenge that remains in the field of anomaly detection is dealing with imbalanced datasets. In many real-world scenarios, anomalous data points may be rare compared to normal data points. This can make it difficult to train machine learning models to accurately detect anomalies. One solution to this problem is to use techniques such as oversampling, undersampling, or data augmentation to balance the dataset [7].

Another challenge is to deal with concept drift, which occurs when the distribution of data changes over time. In the context of anomaly detection, this can result in previously normal data points becoming anomalous, or vice versa. To address this challenge, researchers have proposed methods such as online learning and ensemble methods [8].

C. Problem Definition

PureFacts has access to numerous data points that will be used to calculate the fee that the advisor will charge their investors. If the fee charged is incorrectly calculated, the advisor and the financial firm will be liable, and there might be financial penalties. This anomaly detection system aims to automatically detect errors in fee calculations and produce an alarm system to send push notifications to the advisor for an ongoing fee calculation error. This will help advisors detect errors in advance, saving the institution millions in fines, regulation costs, and compliance issues. Since the fee errors don't happen often, we need to use algorithms that can

detect subtle changes in the data, which are called anomalies or outliers.

The specific anomalies we are detecting are fee schedule anomalies. This anomaly occurs when the fee has been assigned incorrectly, in which case the advisor has mistyped the fee value incorrectly.

II. Methodology

To begin creating a solution, a deep understanding of PureFacts' business model must be made. To make connections in any data received, it must be known how these parameters relate to each other in order to build a model around it. The data was given in multiple files, consisting of hundreds of thousands of data rows. As the data points given are not classified as anomalous/non-anomalous, we must use unsupervised learning approaches. Let us introduce the data parameters that are relevant to the analysis:

- *AssignedBPSRate*: The rate that PureFacts charges a client for advising.
- *CurrentGroupValue*: Numerical value representing the size of PureFacts' client's account.
- *StartTier*: This relates to the starting range of account size.
- *EndTier*: This relates to the ending range of account size.
- *FeeScheduleAssignID*: Represents the group of fee schedules that the row resides in.
- *FeeTypeID*: Represents the type of fee group within the group of fee schedules.
- *FeeGroupID*: This parameter marks data rows that belong to the same fee group.
- *Min/MaxBPSRate*: The minimum or maximum *AssignedBPSRate* that the client can be charged.

As instructed by our client, it was chosen to use the *AssignedBPSRate* and *CurrentGroupValue* to build a model, as these are directly related. As clients' account size increases, the rate they are charged is expected to decrease. The *AssignedBPSRate* determines how much PureFact's clients are billed, so it is imperative that this parameter is not anomalous.

A. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering machine learning model which is great at detecting outliers in an unsupervised manner. This model takes two parameters; an *eps* (epsilon) value, representing the maximum distance between any two points in the same cluster, and a *min samples* value, representing the minimum number of data points in a cluster. The data columns we fed into our model are those most relevant to the *AssignedBPSRate* calculation, such as *CurrentGroupValue*, *FeeScheduleAssignID*, and *FeeTypeID*. These data columns help compare the *AssignedBPSRate* and *CurrentGroupValue* within *FeeScheduleAssignID* subgroups to check for outliers.

To determine the optimal *eps* value within the DBSCAN model, we used metrics from the Silhouette Coefficient (SC) and Davies Bouldin Index (DBI). The SC measures how well each data point fits into its assigned cluster, by comparing its distance to other data points within its own cluster vs. those in neighboring clusters. A higher score indicates better clustering. The DBI measures the average similarity between each cluster and its most similar cluster, while also taking into account the distance between their centroids. A lower value indicates better clustering. Therefore, we chose an epsilon with the highest SC and the lowest DBI.

Since this is an unsupervised machine learning problem, we found it difficult to evaluate our results, as there was initially no clear guideline on what should be flagged as an anomaly.

B. K-Means

K-means Clustering is an iterative approach that fits data into a chosen number of clusters. To begin, k data points are chosen at random to represent initial cluster centroids, where all data points are assigned to their nearst "cluster". The new centroid is then calculated by taking the mean value of all data points in the cluster. Data points are then re-assigned to their closest centroid, and this process is repeated until the centroids no longer move a significant amount. When the algorithm converges, all data points are held in clusters that have similarities among them. To choose a number of clusters, the *elbow method* is used to find the required number before running the K-means algorithm.

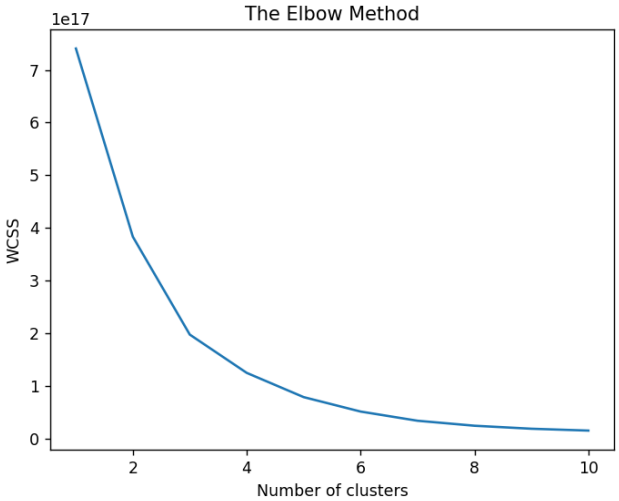

Fig. 1. Elbow method for the dataset.

For the elbow method, the Within-Cluster Sum of Square (wcss) is calculated for a range of cluster amounts. When the curve begins to flatten, the number of optimal clusters is extracted. In Figure 1, this number of clusters is 5, which is then passed into the K-Means algorithm. This model is primarily used to visualise data and get a deeper understanding of parameter relations.

C. Algorithmic Methods

There are two additional ways to identify anomalies without using machine learning models. These algorithmic approaches in this section are used in conjunction with our DBSCAN and K-Means models to discover outliers.

First, anomalies can be identified by comparing the *AssignedBPSRate* value to the *MaxBPSRate* and *MinBPSRate* values. If the *AssignedBPSRate* is outside the *MaxBPSRate* and *MinBPSRate* range, then it can be easily flagged as anomalous.

Second, the *StartTier* and *EndTier* values can give us a window into outliers in the data. If the tiers are increasing, the *AssignedBPSRate* should be decreasing, as the charge for advisors should lower as the value tiers get higher for each fee schedule. Thus, by sorting all rows in a fee schedule group by the *StartTier* and *EndTier* values, we can easily see if the *AssignedBPSRate* is increasing. If so, then we can flag all rows in that fee schedule group as anomalous.

III. Results

The results we attained from our DBSCAN approach, are shown in Figure 2 below. We identified an anomaly frequency within the guidelines from our client but must limit further details due to NDA.

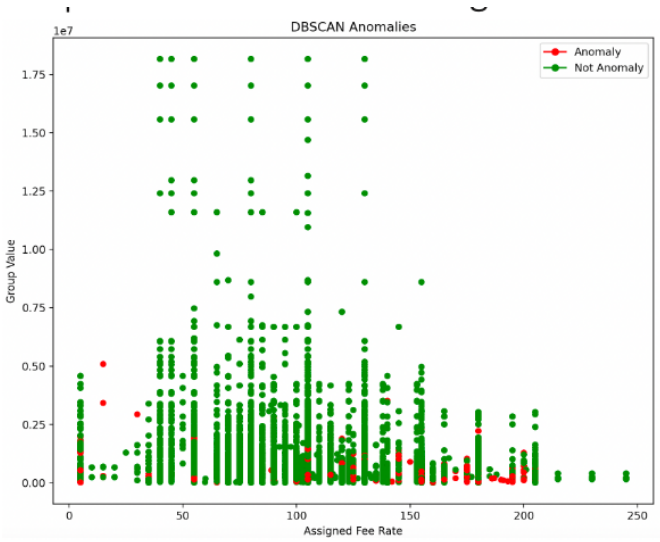

Fig. 2. Results from the DBSCAN model.

Our Algorithmic approaches also identified anomalies, but only a very small percentage of the total anomalous data.

Analysis was also done using the K-Means algorithm, again using *CurrentGroupValue* to find outliers in the *AssignedBPSRate*. The data was grouped into 5 clusters, determined by the *elbow method*. The results obtained are shown in Figure 3 below.

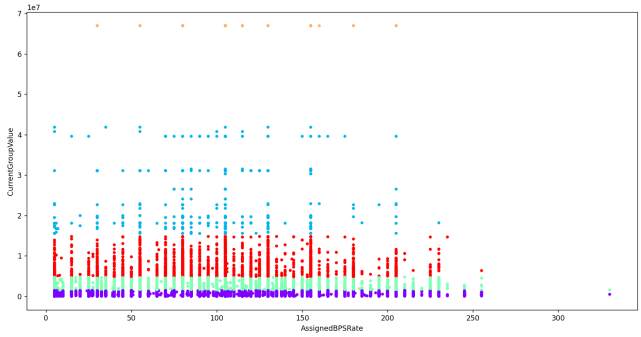

Fig. 3. Results from the K-Means model.

Figure 3 shows 5 clusters stacked on top of each other. This approach showed further insight into how the *CurrentGroupValue* and *AssignedBPSRate* data related to one another. Anomaly detection algorithms using K-Means were explored, however, did not show acceptable accuracy. This model provided a better understanding on how the data was laid out, and propelled us toward finding better approaches of outlier detection.

IV. Conclusion

In conclusion, the anomaly detection system developed for PureFacts Financial Solutions has been successful in detecting anomalies in data records. Our DBSCAN approach identified a realistic number of rows as anomalous, which closely aligns with the guidelines given by our client. Additionally, our algorithmic approaches provided an additional layer of anomaly detection to the system.

Moving forward, it would be important to continue to optimize the system and ensure that all anomalous data is caught. This may involve tweaking the parameters of the DBSCAN and K-Means models to improve accuracy. Additionally, further analysis could be done to identify new types of anomalies and develop methods to detect them.

Another challenge that remains is the scalability of the system. As PureFacts continues to grow and gather more data, the system will need to be able to handle larger and larger datasets. This may require further optimization and potentially the implementation of new technologies.

Overall, the development of this anomaly detection system is a crucial step in ensuring the accuracy of PureFacts' data records and preventing detrimental legal or regulatory implications. Continued work in this area will help to further improve the system and ensure that PureFacts can continue to provide transformational WealthTech solutions to their clients.

REFERENCES

[1] A. B. Nassif, M. A. Talib, Q. Nasir and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review," in IEEE Access, vol. 9, pp. 78658-78700, 2021.

[2] Y. Li, L. Li, and X. Li, "Anomaly detection in time series data using k-means clustering," IEEE Access, vol. 7, pp. 173 389–173 400, 2019.

[3] J. Ren, Y. Zhang, L. Zhang, and J. Yang, "Anomaly detection in network traffic based on DBSCAN clustering algorithm," in Proceedings of the IEEE 6th International Conference on Cloud Computing and Intelligence Systems, pp. 1207–1211, IEEE, 2018.

[4] M. R. Gholami, R. E. Atani, and S. M. S. Sadough, "Anomaly detection using decision trees: A survey," Journal of King Saud University-Computer and Information Sciences, vol. 30, no. 4, pp. 431–448, 2018.

[5] L. Chen, X. Cao, and W. Wang, "Anomaly detection of time series data based on support vector machine," in Proceedings of the IEEE 2nd International Conference on Big Data Analysis, pp. 72–77, IEEE, 2018.

[6] S. S. Suresh and V. K. Govindan, "Anomaly detection using neural networks – A comprehensive review," Expert Systems with Applications, vol. 120, pp. 144–173, 2019.

[7] B. G. Martins, V. L. R. M. Souza, and A. L. F. de Almeida, "Dealing with imbalanced datasets: An overview," in Proceedings of the IEEE 5th International Conference on Data Science and Advanced Analytics, pp. 1–10, IEEE, 2018.

[8] S. Li, S. Li, Y. Li, and C. Li, "Concept drift detection for online anomaly detection: A review," IEEE Access, vol. 7, pp. 34 940–34 950, 2019.

Applying Machine Learning to Bipolar Disorder Categorization Using the Motor Activity Dataset

Rabab Azeem
Queen's University
azeem.rabab@gmail.com

Julien Pierre Chanel
Queen's University
19jpc@queensu.ca

Jingjing Mao
Queen's University
18jm69@queensu.ca

Aria Maz
Queen's University
aria.maz@queensu.ca

Naser AI-Obeidat
Queen's University
21nao3@queensu.ca

***Abstract*—Bipolar disorder, especially bipolar II disorder, is known to have a high suicide and self-harm rate, and a high misdiagnosis rate. This project is an attempt to identify patients with bipolar II disorder among a group of healthy controls and patients with major depressive disorder. We use the motor activity data collected by the motion-sensitive sensor at Haukeland University Hospital, Bergen, Norway. We discuss the preprocessing methods performed and a potential method of feature extraction. We applied a convolutional neural network and a long short-term memory network and the accuracy was 63.48% and 75.77% respectively. Finally, we discussed future directions and suggested methods of improving the accuracy.**

I. Introduction

Beyond the surface of one of the most prevalent mental illnesses lies a harsh reality. Over the past 2 decades, bipolar disorder has resulted in self-harm among 30-40% of patients and led to the suicide of 6% [7]. There are 3 main categories of bipolar disorder, although sometimes their types can blend [5]:

- Bipolar I disorder: is defined by manic episodes that last for at least 7 days (nearly every day for most of the day) or by manic symptoms that are so severe that the person needs immediate medical care [5].
- Bipolar II disorder: is defined by a pattern of depressive episodes and hypomanic episodes. The hypomanic episodes are less severe than the manic episodes in bipolar I disorder [5].
- Cyclothymia: (also called cyclothymia) is defined by recurring hypomanic and depressive symptoms that are not intense enough or do not last long enough to qualify as hypomanic or depressive episodes [5].

Despite its significant consequences and risk, bipolar disorder is commonly known as being hard to be diagnosed. Specifically, Bipolar type II, which has a less severe manic episode, can be mistaken as unipolar depression. Compared to depression and anxiety, the literature about applying machine learning to bipolar disorder diagnosis is rather limited. A literature review paper [4] of bipolar disorder diagnosis and machine learning suggests that bipolar disorder detection is a rapidly improving area of research, even though before 2021 there were only 2 publications from Canada that talked about machine learning and bipolar disorder diagnosis. Among all machine learning techniques applied to bipolar disorder research, the most used were classification models and regression models. Due to the effect of bipolar disorder on daily activities, previous work has been done on applying machine learning algorithms to the motor activity dataset collected from around 50 patients over several days [1]. Further improvements in models and machine learning algorithms were achieved and machine learning has been proven promising in bipolar disorder diagnosis [6] [3]. This project aimed to address the misdiagnosis of Bipolar type II and major depressive disorder. Therefore, we focused on the classification of healthy controls, bipolar II disorder and major depressive disorder.

II. Methodology

A. Introduce the dataset

The proposed data set is the Depresjon dataset [1] which consists of motor activity recordings of 23 unipolar and bipolar depressed patients and 32 healthy controls at Haukeland University Hospital, Bergen, Norway [1]. The activity level was recorded with an actigraph watch worn that can detect movements over 0.05g at the patient's wrist. The sampling frequency of the actigraph watch is 32Hz, and the output activity level is saved every minute.

The paper associated with the Depresjon dataset used machine learning to classify patients into depressed and non-depressed. Consisting of two folders, the dataset contained data for the control and condition groups. Each patient had a designated csv file containing timestamped data (in minutes) of physical activity, date of measurement, and activity measurements collected from their actigraph watch. Additionally, vital patient-specific data were included such as afftype (bipolar II, unipolar depressive, bipolar I), melanch (melancholia, no melancholia), gender, age groups, education levels, marriage status, and MADRS scores at the start and end of the measurement period. MADRS scores are a rating scale widely used and validated tool for measuring the severity of depressive symptoms in individuals [2].

B. Data preprocessing

There are mainly three steps to clean the data. The only demographic features available for healthy controls are gender and age, meaning if we want to include healthy controls in the training dataset, we need to drop all the demographic columns except for age and gender. Secondly, we transformed the raw dataset. For the activity recordings of each patient, we extract the recording of each day, and attach the demographic information of the corresponding patient after the daily

recording. Therefore, each row of the output dataset is the 24hr recording of a patient with the patient's demographic info attached. During our preprocessing, we also realized that some healthy controls forgot to wear their actigraph watch after several days of recording. Therefore, we dropped days with too many identical data points.

Our analysis aimed to establish the correlation between physical activity and mental health using both statistical learning and deep learning techniques. To achieve this, we leveraged the vast collection of motor activity signals that were timestamped every minute for a large number of patients. However, due to the sheer volume and complexity of the data, accurate model building necessitated feature selection. Therefore, we plan to apply a feature selection process to identify the most relevant features in the motor activity recordings, which could assist in generating accurate MADRS scores and diagnoses. To simplify the data for machine learning algorithms, we transformed the daily numerical motion data into categorical labels consisting of 0, 1, 2, or 3. The labels were determined based on the average motion within 10-minute intervals. A label of 0 indicates that the averaged motion was 1 standard deviation below the daily average, and the non-zero values of the original motion data within the 10-minute interval were less than 40%. A label of 1 indicates that the averaged motion was 1 standard deviation below the daily average, but the non-zero values of the original motion data within the 10-minute interval were more than 40%. A label of 3 indicates that the averaged motion was 1 standard deviation above the daily average. A label of 2 is used for all other cases. By categorizing the data in this way, we hope to make it easier for machine learning algorithms to interpret the data and identify patterns. Due to time constrain, we didn't use the proposed data transformation method in our training dataset.

C. Models

A Multi-Layer Perceptron (MLP) is a artificial neural network which consists of different layers of nodes that process data, and each node is connected to all nodes in the previous layer. The nodes in the input layer of the MLP are representative of the attributes of the data, and the nodes in the output layer are representative of the predicted output. In between these two layers is the hidden layer, which is responsible for executing activation functions such as the sigmoid, tanh (hyperbolic tangent), and the ReLu (Rectified Linear Unit). Activation functions allow neural networks to establish nonlinear relationships in data, and this is crucial as the vast majority of real world problems cannot be modeled by linear functions. During a process called backpropagation, the weights in the MLP are adjusted using an optimization algorithm called gradient descent to minimize the error found in the predicted output. CNNs use specific layers such as pooling and convolutional layers to extract features from images. Consequently, the number of parameters necessary for training is reduced, leading to a more efficient pattern recognition model. Knowing the power of pattern recognition of CNN, we deduce that CNN may be able to find the difference between depression and bipolar disorder since the activity level of a patient with bipolar disorder tends to have a large fluctuation within a day.

Long short-term memory (LSTM) networks were developed to solve the problem of processing long-term dependencies in sequential data, such as speech or text. Contrary to MLPs, LSTMs use memory gates and cells that enable them to retain or ignore data from earlier time steps in a sequence. We think LSTM is a reasonable choice because the time series nature of the dataset and the time of activity level reflects the patient's circadian rhythm. People with mental health disorders tend to have sleep difficulties and different circadian rhythms from healthy controls. Here are the model architectures of each model:

1) Convolutional neural network:

- 1 dimensional convolutional layer
- 1 dimensional convolutional layer
- dropout
- 1 dimensional Pooling using maximum
- flattening
- dense layer
- dense layer

2) Long short-term memory network:

- LSTM layer
- dropout
- dense layer
- dense layer

D. Results

The accuracy of CNN was 63.48%, and the accuracy of LSTM was 75.77%. The accuracy of LSTM is higher, meaning the circadian rhythm of the patients contributes to the diagnosis of bipolar disorder, and future modelling should take the time series nature of the dataset into account.

III. Conclusion

In conclusion, the performance of CNN and LSTM without feature extraction and further preprocessing was limited. The lack of reliable datasets is the biggest difficulty in applying machine learning algorithms to mental health disorder detection. The number of data entries of the dataset we used is less than 1k after cleaning, which may limit the model from being exposed to a more diverse dataset and recognizing the more general pattern. The lack of data points can also lead to bias against underrepresented groups in the chosen dataset. Moreover, extracting daily activity has a risk of erasing the long-term pattern of the dataset and reducing the amount of information the model can interpret since mental health disorders can only be diagnosed after days of symptom monitoring. Hence, we suggest future studies consider the long-term effect of mental health disorders. Also, after extracting the daily activity level of each patient, the training and testing data points may contain the activity level data from the same patient, which may cause overfitting. We also suggest using data with heart rate monitoring or other physiological information to assist in the interpretation of activity levels.

In real life, the portion of the population with mental health disorders is relatively small, meaning the technique of dealing with an imbalanced dataset needs to be used, for instance, SMOTE. It is also worth trying to apply the proposed data transformation method to the cleaned dataset. One can use the transformed data with some statistical methods such as logistic regression, classification tree, and random forest. If the transformed dataset erased too much information, one can also combine the transformed dataset with the original dataset. A more in-depth validation of the result is needed. For instance, the model may be a lot less sensitive to a certain type of disorder. When the number of data points is small, we suggest a Leave-One-User-Out validation strategy.

Overall, deep learning algorithms are promising in bipolar disorder diagnosis, but we face the challenge of the lack of datasets and the long-term effect of mental health disorders.

REFERENCES

[1] Enrique Garcia-Ceja, Michael Riegler, Petter Jakobsen, Jim Tørresen, Tine Nordgreen, Ketil J. Oedegaard, and Ole Bernt Fasmer. Depresjon: A motor activity database of depression episodes in unipolar and bipolar patients. In *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, page 472–477, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Wolfgang Hiller, Gabriele Dichtl, Heidemarie Hecht, Wolfgang Hundt, Werner Mombour, and Detlev von Zerssen. Evaluating the new icd-10 categories of depressive episode and recurrent depressive disorder. *Journal of Affective Disorders*, 31(1):49–60, 1994.

[3] Petter Jakobsen, Enrique Garcia-Ceja, Michael Riegler, Lena Antonsen Stabell, Tine Nordgreen, Jim Torresen, Ole Bernt Fasmer, and Ketil Joachim Oedegaard. Applying machine learning in motor activity time series of depressed bipolar and unipolar patients compared to healthy controls. *PLOS ONE*, 15(8), 2020.

[4] Zainab Jan, Noor AI-Ansari, Osama Mousa, Alaa Abd-alrazaq, Arfan Ahmed, Tanvir Alam, and Mowafa Househ. The role of machine learning in diagnosing bipolar disorder: Scoping review. *Journal of Medical Internet Research*, 23(11), 2021.

[5] National Institute of Mental Health. Bipolar disorder. https://www.nimh.nih.gov/health/topics/bipolar-disorder. Accessed: 2023-03-01.

[6] Praveen Manoj Singh and P. S. Sathidevi. Design and implementation of a machine learning-based technique to detect unipolar and bipolar depression using motor activity data. *Lecture Notes in Networks and Systems*, page 99–107, 2021.

[7] wikipedia. Bipolar disorder. https://en.wikipedia.org/wiki/Bipolar_disorder. Accessed: 2023-03-01.

Atlas: Find Anything on YouTube

Tomiwa Ademidun
Atila Inc.
tomiwa@atila.ca

Aaron Doerfler
Ivey Business School
adoerfler.msc2023@ivey.ca

Shania Sheth
Queen's University
shania.sheth@queensu.ca

***Abstract*—Atlas uses OpenAI's Whisper, UKP Labs Bidirectional Encoder Representations from Transformers (BERT's) sentence transformer model, and bidirectional and auto-regressive transformers (BART) long-form question answering (LFQA) system to summarize and search any YouTube video. Atlas is designed to improve access to information and help create a more equitable and informed society through the use of artificial intelligence solutions. It takes a transcript and breaks it up into smaller segments converted into a 768 vector array. The vector array is saved in a vector database provided by Pinecone and then uses this database to determine which transcript segment vector is nearest to our search phrase vector and combines the search results to create a long-form answer that is outputted to the user. Atlas saves a significant amount of time and resources by summarizing one of the world's largest hubs of knowledge.**

I. Introduction

Atlas is an AI-powered search engine that can summarize and search any YouTube video. YouTube is one of the world's largest sources of information with about 500 hours of video uploaded to YouTube every minute [1]. However, unlike Reddit, Twitter, Wikipedia, books and other text-based information sources, information on YouTube is not well indexed. Sure, you can do a keyword search but can't find the precise timestamp of the information you want. Atlas fixes this. A search engine for one of the world's largest hubs of knowledge makes it easy to access a huge untapped source of information. The user enters the URL of any YouTube video and a query term, and Atlas will return terms directly related to the search query. It works by getting the transcript of a YouTube video using the URL from the YouTube transcript API or if that transcript does not exist, it downloads the audio of the video as an mp3 file with Pytube and uses OpenAI Whisper to transcribe. Then, the transcript is broken up into shorter segments which are converted into a 768 vector array using UKP Labs Bidirectional Encoder Representations from Transformers (BERT's) sentence transformer model. Atlas then uses Pinecone to save the vector array in a vector database and takes the search phrase and embeds it into a 768 vector array. Using the vector database Atlas determines which transcript segment vector is nearest to our search phrase vector and combines the search results to create a long-form answer using bidirectional and auto-regressive transformers (BART) long-form question answering (LFQA).

A. Motivation

In the last few years, we have seen a rise in AI research projects related to language modelling. Most recently in September of 2022, OpenAI released their open-source automatic speech recognition (ASR) neural net called Whisper [2]. In the announcement blog post which summarizes the research paper, OpenAI says that the decoder is trained to generate the text caption that corresponds to the audio, while also incorporating special tokens that guide the model to perform additional tasks like identifying language, creating phrase-level timestamps, performing multilingual speech transcription, and translating speech to English (see Figure 1) [3]. The developers envisioned a wider set of applications related to voice applications, and while YouTube does try and caption each of their videos, Whisper can do it in real-time, faster, and with much better performance.

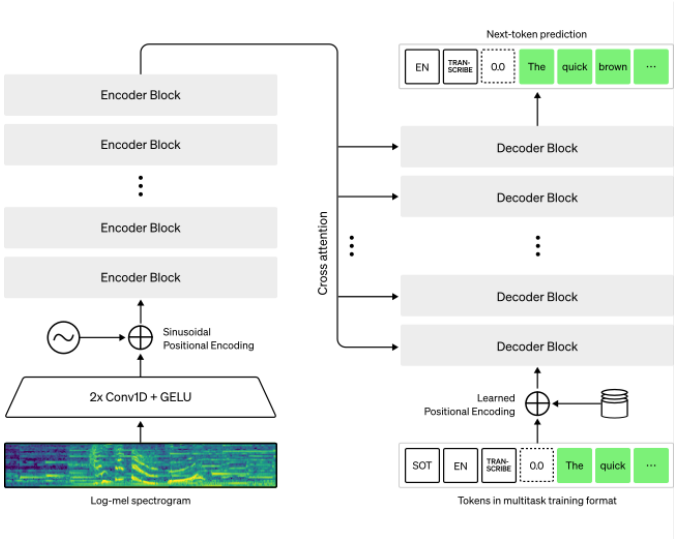

Fig. 1. OpenAI Whisper diagram, sourced from their research paper.

Sentence-BERT, a modification of the pre-trained BERT network that uses siamese and triplet network structures (see Figure 2) [4], is another recent AI research project in language modelling. SBERT can be leveraged to generate high-quality sentence embeddings, which can be used to identify the most semantically relevant sentences in a video's transcript. By identifying these key sentences, an AI video summarization tool could automatically generate a summary of the video's content that captures the most important aspects of the video. This has the potential to save time and effort for students, researchers, and anyone who needs to quickly understand the

key points of a video without having to watch the entire thing. Thus, we thought using this summarization tool could significantly enhance the accuracy and efficiency of the tool we intended to create.

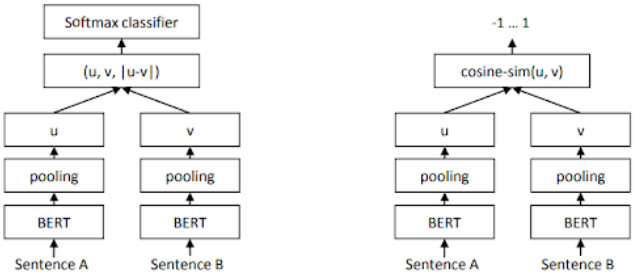

Fig. 2. SBERT architecture with classification objective function and SBERT architecture at inference, for example, to compute similarity scores, sourced from their research paper.

The last piece of the puzzle, so to speak, comes in the form of question-answer systems. In 2019, Angel Fan et al wrote their research paper titled "ELI5: Long Form Question Answering". This sequence-to-sequence (seq2seq) system took 270,000 question-and-answer pairs from the subreddit "Explain Like I'm Five", which is exactly as the title describes, as data, and use it to output paragraph-length explanations in response to complex questions (see Figure 3) [5]. This would allow us to create something that could make complex topics more digestible and more concise for readers.

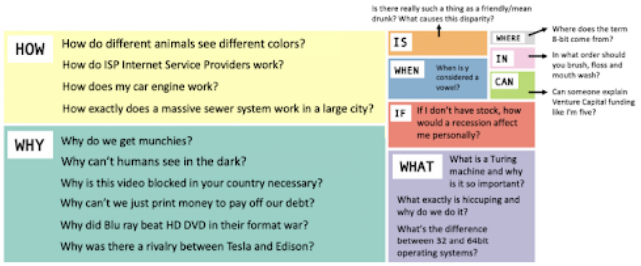

Fig. 3. ELI5 questions by starting word, where box size represents frequency, sourced from their research paper.

Using these three AI technologies we decided that we wanted to explore ways to make information on the internet more digestible for the average user.

B. Related Works

One person we got a lot of inspiration from was James Briggs. James is a freelance machine learning engineer, startup advisor, and developer advocate at Pinecone. James has published blog posts and YouTube tutorials using both Whisper (see Figure 4) [6] and Natural Language Processing [7] to revamp YouTube searches. James' methodology served as the basis for Atlas. What he did is create an app that can get specific timestamps that answer the search query, however, limited currently to 5 channels that are related to ML. His first step was to download YouTube video data and extract the audio in each video using PyTube and then use Whisper to transcribe the audio. Using Hugging Face datasets that include short clips of the transcribed audio, it merges every six segments while also moving forward three segments to make sure that meaningful segments do not get cut completely and are instead included in the next iteration. Using a Hugging Face QA model from their transformers and sentence transformers library he encodes the segments and inserts the embeddings into a vector database. Then he encodes the query using the same embedding model he used to encode the segments and passes the query to the vector database which returns the relevant information from the transcript and a timestamp from the videos. One limitation of this work is that the current scope is limited to only a few videos talking about machine learning, natural language processing, and vector search. That leaves an area for someone to create something similar with expanded data sets that would allow for any phrase to be searched in any video on YouTube.

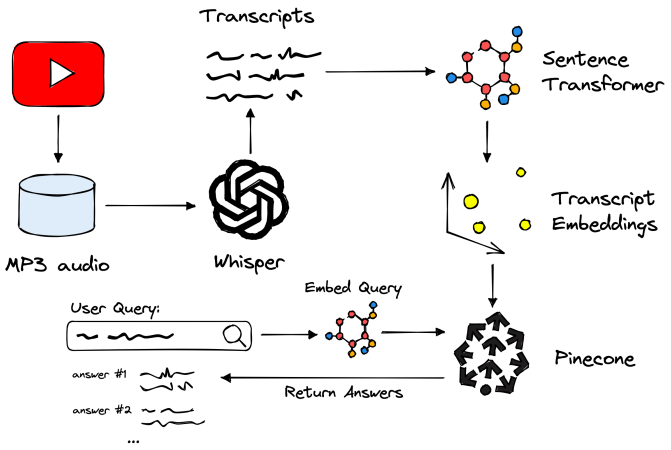

Fig. 4. Overview of the process used, covering OpenAI's Whisper, sentence transformers, and the Pinecone vector database, sourced from Briggs' blog post.

C. Problem Definition

What we see in the blog post from James Briggs is a tool that worked only for five channels and for specific prompts related to ML. We see an opportunity to expand this to include not just a few channels and topics, but unlock the full range of topics that can be found on YouTube. At the most basic level, Atlas summarizes and makes it much more convenient to find specific information in a YouTube video. This could be useful for students, researchers, or just casual viewers who want to learn more about a specific topic more efficiently. However, Atlas also helps makes information more accessible by providing simple summaries of even the most complex terms. This has huge value in improving education access. There are about 5.16 billion people who have access to the

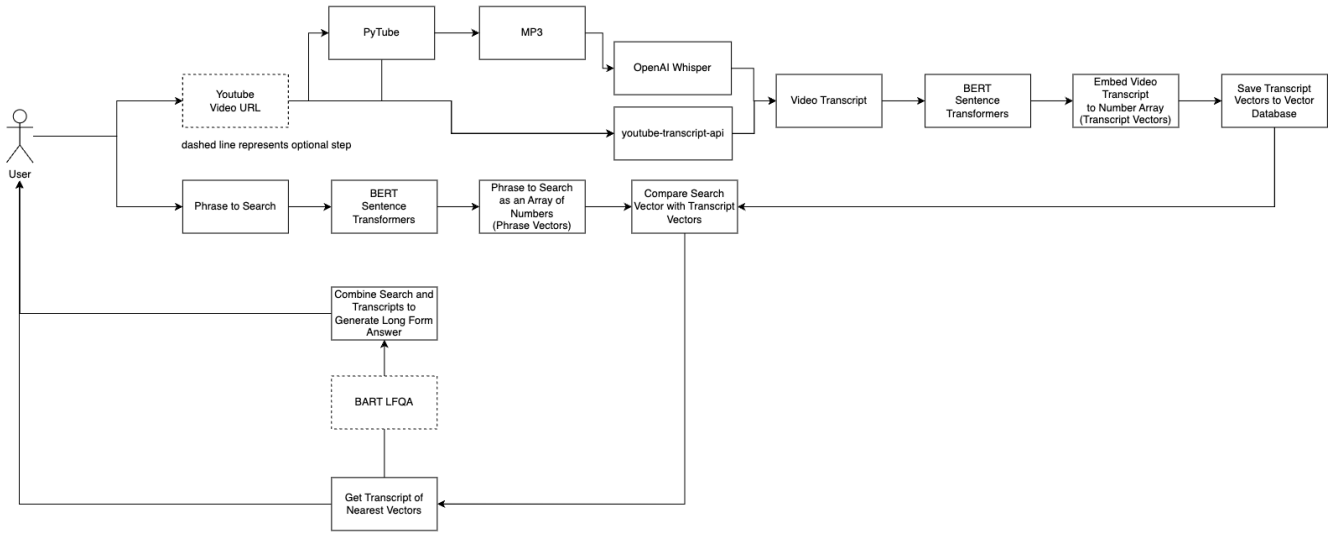

Fig. 5. Full Atlas process breakdown

internet [8] but come from different countries with different languages, cultural norms and educational backgrounds. Atlas' summarization feature makes it easy for someone to learn about something as complex as quantum computing in a much more digestible manner. Even if only a small part, our goal with Atlas is to improve access to information and help create a more equitable and informed society.

II. Methodology

In this section, describe your overall design process, and walk through the details of each step.

The start of the design process stemmed from the James Briggs blog post mentioned in section 1b titled Related Works. For Atlas to work it needs to obtain some sort of transcript from the input YouTube video. The first method of obtaining this video utilizes the YouTube transcript API to extract the transcript by using its URL. This involves sending an API request to the YouTube servers and receiving a response containing the transcript data in a standardized format such as JSON. If the video does not have a transcript available, a second method is needed to download the audio of the video as an MP3 file using the Pytube library and apply Whisper to transcribe it. It is necessary to break up an audio transcript into shorter segments to facilitate further analysis, so we used UKP Labs BERT's sentence transformer model to transform each segment into a 768-dimensional vector array. This model is a variant of the BERT architecture that has been pre-trained on large amounts of text data and can effectively capture the semantic meaning of sentences [9]. The process of converting each segment to a vector array involves passing the segment through the sentence transformer model, which outputs a 768-dimensional vector that represents the semantic meaning of the segment. We then needed some sort of vector database to save the vector array and decided to go with Pinecone's cloud-based vector database service. To begin you have to create a Pinecone client object, which connects to the Pinecone API. The vector array is indexed using a unique identifier that allows it to be easily retrieved later. This involves calling the Pinecone index method, which takes the identifier as input and the vector array to be stored. Pinecone then automatically encodes and compresses the vector data and stores it in a distributed, high-performance storage system. Once the vector array has been indexed, it can be easily queried using Pinecone's search API, which takes as input a query vector and returns the most similar vectors in the database [10].

When the user enters a query, it is then embedded into the 768 vector array and using the vector database, Atlas finds which transcript segment vector is nearest to our search phrase vector. Two vectors are considered to be "near" or "similar" if they are close to each other in terms of some distance metric and are often euclidean distance or cosine distance, which measures the geometric distance or the angle between the vectors [11]. This indicates that these vectors share similar properties. Finally, Atlas used BART LFQA to create a long-form answer. BART is a pre-trained language model that has shown excellent performance in answering complex questions in a more digestible manner. For example, the user enters a search term such as "what shoes should I wear" and it returns a list of matches. Then we take each of those matches and ask our generator model to combine them to generate a coherent sentence (see Figure 5 for a visual depiction of the full process).

There are a number of ways to evaluate the success of Atlas. One such metric is the precision and recall of the summary, which measures the percentage of important information that is included in the summary and the percentage of unimportant information that is excluded. As we continue to use and test Atlas we also have a part to play in assessing the summary quality. We hope to have users read over the summary and rate its overall effectiveness in conveying the key information from the video and provide us with feedback. There could be other methods too, such as the diversity and coverage of the summary. In terms of finding terms related to the search query, being sure that those are indeed related and that Atlas has picked out the most related terms would be useful in evaluating the success as well.

III. Results

We found that the project does save a significant amount of time and resources by quickly providing summaries of long videos and presenting relevant information to the user's search query. For example, we tested out a clip from the Joe Rogan Experience where he is interviewing AI scientist Lex Fridman [12] with the search query "basketball". Even though the word basketball is never said, Atlas returned the exact timestamp where Lebron James and Michael Jordan are brought up in the conversation. Additionally, when we ask Atlas to summarize a closet organization video [13], it returns "A wardrobe essentials video is being made by a YouTube channel. The purpose of the video is to give a more comprehensive guide to how to pick a flattering fit and good quality pieces so that you can build a solid base to your wardrobe and then branch out from there." Thus, it seems that Atlas is fairly accurate in its output. In the future we hope that it could provide the name of the creator, for example, to make the summary even more accurate.

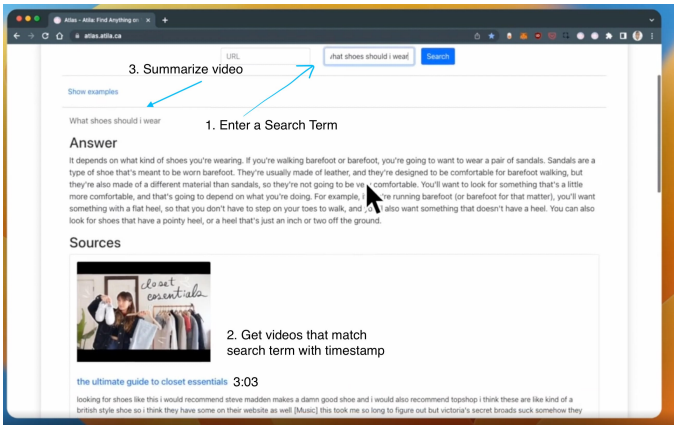

Fig. 6. Atlas closet example

Over the course of this project, we got to learn about the different ways to gather the information included in videos and how these can be effectively summarized using various ML techniques. This includes getting a better understanding of how much more effective Whisper is compared to YouTube's built-in transcription software. We also got to explore different language models such as BERT and BART LFQA and how those AI applications could help us work towards our goal. These integrations have so far worked as expected which has not forced us to go look for other alternative solutions. The biggest advantage of using this method was that we are getting technology that does exactly what we need it to do. This saved us a lot of time by not forcing us to go and build our own transcriber or sequence-to-sequence system but instead focusing on putting them together and creating the user interface. The hard part was deploying an ML app to a production environment so that other people could use it. This was frustrating because we would spend an entire day trying to use more "do it yourself" hosting options like Sagemaker only to realize that it was too complicated and have to then switch to a different provider. However, it forced us to learn about a lot of very interesting ML deployment products like banana.dev and Paperspace even though they all ended up being too confusing to use. For example, there is a way to get whisper to include the timestamp when you transcribe but you need to pass in the argument verbose=true. The models that claimed to be "easy to use" achieved this easily by removing a lot of customizability, so It was not clear that any of the three options we tried would allow us to add timestamps. All of the ML deployment options we came across were unusable until we came across the Hugging Face Inference endpoint which provided the perfect mix between Sagemaker's customizable but complicated setup and the cottage industry of other providers where you could not customize anything.

IV. Conclusion

To conclude, Atlas is an AI-powered search engine that can summarize and search any Youtube video. By making a search engine for one of the world's largest hubs of knowledge, we unlock access to a huge untapped source of information. Atlas makes it convenient to find specific information and could be useful for students, researchers, or just casual viewers who want to learn more about difficult topics in a digestible manner. The search function also helps users find the most relevant information to their specific search query.

The great thing about the current state of AI is that a lot of resources are open-sourced. This is very useful in helping people learn how to code or being able to utilize APIs in their own work. Model creation and implementation are very advanced however, model deployment still has a lot of room for improvement. One immediate issue we have to solve is the feasibility of hosting services. Hosting our model on Hugging Face costs 438.05 a month and we use a Small GPU 1X Nvidia Tesla T4. Huggingface does not support updating existing endpoints to a newer commit so each time we update handler.py we had to delete existing endpoints and deploy a new one. We updated the code frequently and created newer endpoints, but forgot to delete the older ones and thus we had 4 endpoints running at the same time costing us an extra $100.Paying400$ is not

necessarily the problem, but our service is still a relatively small side project serving less than 100 requests a day so $400 for such low usage is way too much. In the future, we also want to make sure that Atlas improves the quality of its summary responses$

Atlas is an open platform. We chose this because we believe that open platforms last longer than closed platforms and are generally better for society. Such a powerful tool should be built in the open to force transparency and make decisions that are most aligned with humanity. Atlas is a stepping stone on this path, and we are excited to play a role in improving access to information and education through the use of artificial intelligence.

REFERENCES

[1] L., C. (2022, June). Youtube: Hours of video uploaded every minute 2022. Statista. Retrieved March 2, 2023, from https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/

[2] Radford, A., Kim, J. W., Xu, T., Brockman, G., Mcleavey, C., amp; Sutskever, I. (2022, September 21). Robust speech recognition via large-scale weak supervision. OpenAI. Retrieved March 2, 2023, from https://cdn.openai.com/papers/whisper.pdf

[3] Open AI. (2022, September 21). Introducing Whisper. OpenAI. Retrieved March 2, 2023, from https://openai.com/research/whisper

[4] Reimers, N., amp; Gurevych, I. (2019, August 27). Sentence-BERT: Sentence embeddings using Siamese Bert-Networks. TU Darmstadt. Retrieved March 2, 2023, from https://arxiv.org/abs/1908.10084

[5] Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., amp; Auli, M. (2019, July 22). Eli5: Long form question answering. arXiv.org. Retrieved March 3, 2023, from https://arxiv.org/abs/1907.09190

[6] Briggs, J. (n.d.). OpenAI whisper: Introduction and example project. Pinecone. Retrieved March 2, 2023, from https://www.pinecone.io/learn/openai-whisper/

[7] Briggs, J. (n.d.). Making YouTube search better with NLP. Pinecone. Retrieved March 2, 2023, from https://www.pinecone.io/learn/youtube-search/

[8] Petrosyan, A. (2023, February 24). Internet and social media users in the world 2023. Statista. Retrieved March 2, 2023, from https://www.statista.com/statistics/617136/digital-population-worldwide/: :text=As

[9] Reimers, N., amp; Gurevych, I. (2019, August 27). Sentence-BERT: Sentence embeddings using Siamese Bert-Networks. TU Darmstadt. Retrieved March 2, 2023, from https://arxiv.org/abs/1908.10084

[10] Pinecone. (n.d.). Overview. Pinecone. Retrieved March 2, 2023, from https://docs.pinecone.io/docs/overview

[11] Briggs, J. (n.d.). Using semantic search to find gifs. Pinecone. Retrieved March 2, 2023, from https://www.pinecone.io/learn/gif-search/search

[12] Rogan, J., amp; Fridman, L. (2022, December). Joe Rogan amp; Lex Fridman: Lionel Messi Is The GOAT Over Cristiano Ronaldo. YouTube. Retrieved March 2, 2023, from https://www.youtube.com/watch?v=cnFCGOWHQ7A

[13] bestdressed. (2019, April 18). The Ultimate Guide to Closet Essentials. YouTube. Retrieved March 3, 2023, from https://www.youtube.com/watch?v=TuPyVPdH814

Benchmarking Training and Inference Times of Deep Learning Frameworks in Python and Julia

Trevor Yu
University of Waterloo
trevor.yu@uwaterloo.ca
Author

Anusha Raisinghani
University of Waterloo
araising@uwaterloo.ca
Contributor

Musaab Siddiqui
University of Waterloo
mm4siddi@uwaterloo.ca
Contributor

Urban Pistek
University of Waterloo
upistek@uwaterloo.ca
Contributor

Yash Pokra
University of Waterloo
ypokra@uwaterloo.ca
Contributor

Adish Shah
University of Waterloo
ap8shah@uwaterloo.ca
Contributor

Ethan Gabriel
University of Waterloo
e6gabrie@uwaterloo.ca
Contributor

***Abstract*—While the two most popular deep learning frameworks in 2022 were TensorFlow and PyTorch, there are many other actively developed, open source frameworks available in the Python and Julia programming languages. In this paper, we build models from scratch using six frameworks: TensorFlow, PyTorch, MXNet, and JAX in Python, and Flux and KNet in Julia. We train a multi-layer perceptron on the MNIST dataset and a ResNet on the CIFAR-10 dataset. From these tests, JAX had the fastest runtime metrics for training on all tasks, being 8.5x - 17x faster than other frameworks for ResNet epoch training times, and despite also including the compile time, had the fastest total training times. Although JAX had the fastest runtime, writing JAX code is less developer friendly than other popular Python frameworks. In future work, we plan to benchmark more models such as a transformer.**

I. Introduction

A. Motivation

In recent years, advances in AI models such as GPT-3 Stable Diffusion have been based on deep neural networks. To build and train neural networks, there have been many deep learning frameworks developed in various programming languages. According to the Stack Overflow Developer Survey in 2022, the two most popular deep learning frameworks are TensorFlow [1] and PyTorch [2], both of which have APIs in the Python programming language [3]. However, there are also many actively developed, open source frameworks available, such as Apache MXNet [4] and Google JAX [5] with Python APIs, and Flux.jl [6], [7] and KNet.jl [8] in Julia. As the ecosystem of deep learning packages has matured and evolved, there have not been any recent, comprehensive comparisons between all these frameworks, particularly on the speed of these frameworks.

B. Background

In deep learning, neural networks are trained by applying first-order gradient-based optimization to update the parameters of the network to minimize a loss function between model outputs and training data labels. The main computational overhead during training a neural network comes from computing the gradients of the parameters through backpropagation.

Automatic differentitation (AD) allows the automatic computation of derivatives of arbitrary numerical functions without having to define exact, symbolic expressions for derivatives of custom functions. AD is implemented by recording function calls in a computational graph and applying the chain rule to low-level operations to compute derivatives [9]. Backpropagation is a special case of reverse-mode AD, which involves a forward pass through a model to record all operations, and a reverse pass to compute the gradient of the loss with respect to each model parameter. Automatic differentiation is core part of deep learning frameworks, though AD can be used for other applications beyond deep learning.

Julia is a flexible, multi-paradigm, dynamic language built with the performance of numerical computing in mind [10]. Julia uses just-in-time (JIT) compilation using LLVM to achieve performance comparable to traditional statically-typed languages. Packages in Julia, like Flux and KNet, leverage the extensible of the language and are essentially all high-level Julia code, which when JIT compiled, gain the performance benefits of Julia. Due to this extensibility, it is possible to integrate multiple different AD engines to compute gradients of neural network models, although the one used by Flux is Zygote, which uses source code transformation to run AD [11]. Because of the JIT compilation, the computation graph generated in the Julia frameworks is static. Julia also has a CUDA.jl package that allows for compatibility of running array computations on Nvidia graphical processing units (GPU) hardware accelerators.

The Python packages for TensorFlow, PyTorch, and MXNet all are high-level APIs in Python to call lower-level, compiled primitives written in C++ and CUDA. These three frameworks all make use of their own tensor array class and have an dynamic, graph-based AD engine that allows for a simple API call for computing gradients via backpropagation. Although the core compiled libraries of these frameworks are fast and ef-

ficient for the hardware, there is still a large speed and memory overhead of using the Python interpreter for defining the high-level code execution. TensorFlow, PyTorch, and MXNet all have modules defining common neural network layers, through `tensorflow.keras`, `torch.nn`, and `mxnet.gluon` respectively.

JAX is an automatic differentiation library that uses the accelerated linear algebra (XLA) engine to JIT compile Python functions to optimized hardware kernels. Unlike the other Python frameworks, JAX uses a static, compiled computation graph of XLA operations for its AD engine. The Flax library [12] is built on top of JAX to provide common neural network layers and an API similar to TensorFlow for building models.

C. Problem Definition

In this paper, we benchmark the speed of various deep learning frameworks from both the Python and Julia programming languages. In Python, we evaluate TensorFlow, PyTorch, MXNet, and JAX. In Julia, we evaluate Flux and KNet. We test these frameworks on two deep learning benchmark tasks: a multi-layer perceptron (MLP) model trained on the MNIST dataset and a ResNet convolutional neural network (CNN) model trained on the CIFAR-10 dataset. We expect that one of the JIT compiled frameworks (JAX, Flux, KNet) will have the fastest runtime metrics.

II. Methodology

Each model is written in each framework, following the specifications laid out in each task. The relevant neural network primitive layers are used from each framework for model creation following standard conventions for that framework. When a layer is not available, a custom implementation is written. At the time of writing, the KNet implementation for the CNN was unable to be completed, so it is omitted from the CNN results.

During testing, each model is trained with 10 repeats, using a different random seed for each repeat. We record task metrics for final training loss and final evaluation accuracy as comparison to ensure models trained in the same way. The runtime metrics collected are the total training time, average epoch training time, and average batch inference time. The total training time includes model forward and backward passes, JIT compile time, and inference on the evaluation dataset after each training epoch. The average epoch training time includes just the model forward and backward passes on the training data. If the framework has JIT compile time, the time of the first epoch is excluded when computing the average epoch training time. The average batch inference time is the total inference time on the test dataset divided by the number of batches in the test dataset. We use the same batch size in training as in inference.

Code for the models, training scripts, and environment specifications is available at https://github.com/WAT-ai/DL-framework-comparison.

A. Multi-layer Perceptron

The first benchmark task we evaluate is the performance of a single hidden layer MLP on the MNIST handwritten digits dataset. This dataset consists of 60,000 28x28 images of handwritten digits from 0 to 9 in greyscale. The images are preprocessed by normalizing the values between 0 and 1 and flattening the images into a 784 dimensional vector. The standard MNIST train split of 50,000 images is used for training and the 10,000 test split is used for evaluation. Targets are converted into one-hot vectors.

The model consists of two dense layers with a ReLU hidden activation. The first dense layer has input size of 784 and output size of 100. The second dense layer has input size of 100 and output size 10, for the number of classes. We use cross entropy loss from logits as the loss function and the Adam optimizer with a learning rate of 1e-3, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The model is trained for 10 epochs with a batch size of 128.

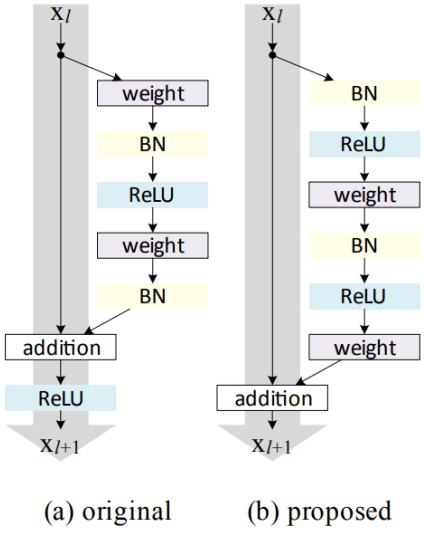

Fig. 1. Comparison of (a) original ResNet and (b) ResNetV2 blocks [13]

B. Convolutional Neural Network

The second benchmark task we evaluate is the performance of a small ResNet on the CIFAR-10 tiny images dataset. This dataset consists of 60,000 32x32 RGB images over 10 different classes of vehicles and animals. The images are preprocessed by first converting int8 values to the range of [0, 1], then normalizing by the ImageNet statistics: subtracting means $[0.485, 0.456, 0.406]$ and dividing by standard deviations $[0.229, 0.224, 0.225]$ for the red, green, and blue channels respectively. The standard train split of 50,000 images is further divided into 45,000 training images and 5,000 validation images. The 10,000 test images are only evaluated after training.

Data augmentation is applied to the training data only as in [14]. The augmentation applied is 4 pixel padding to each side

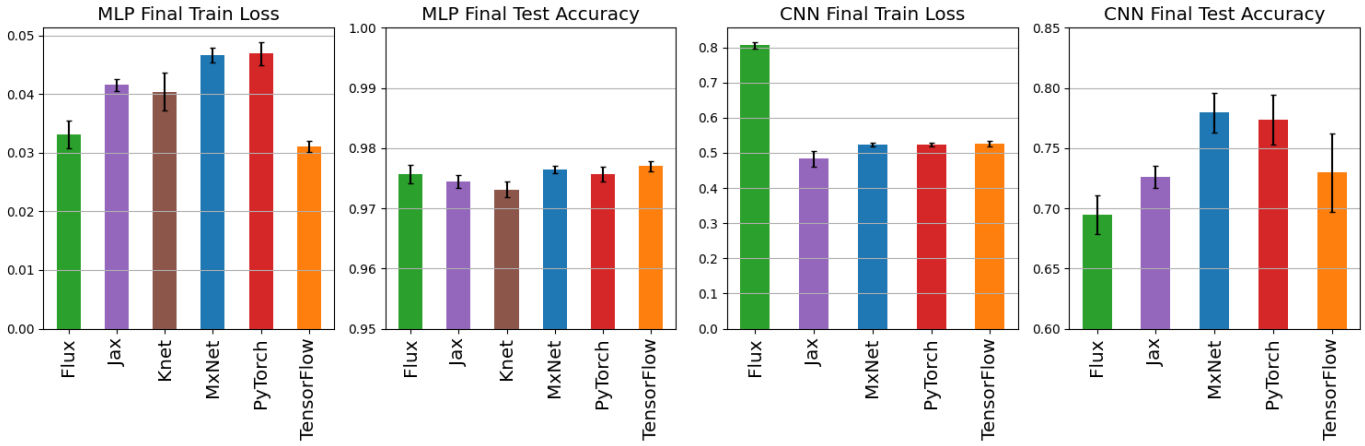

Fig. 2. Task metrics for MLP and CNN models

of the image, applying a random horizontal flip, and applying a random 32x32 crop.

The model is a 20-layer ResNet composed of basic blocks as used in [15] for testing on CIFAR-10, except we used the basic block structure of ResNetV2 [13]. These blocks apply layers in the order of BatchNorm-ReLU-Conv (Figure 1b) of Conv-ReLU-BatchNorm (Figure 1a). The residual used by the model in [15] is the identity shortcut, which downsamples inputs by the same stride factor as the first strided convolution, and fills extra channels in the residual are filled with 0s to ensure the residual shape matches the final shape of the processing layers.

We use cross entropy loss from logits as the loss function and the Adam optimizer with a learning rate of 1e-3, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The model is trained for 10 epochs with a batch size of 128 on a CUDA capable GPU. While this optimization routine is different from [15], the goal is not to recreate the results of another paper, but to benchmark the speed of these different frameworks using a consistent architecture and training method.

C. Hardware and Environment

The computer used for experiments has an Intel i7-3930K CPU, 42 GB of RAM, and a Nvidia Titan Xp GPU with 12 GB of VRAM, using Nvidia driver version 515.85.01. The environments were built in Docker with Nvidia CUDA base images using Ubuntu 20.04, CUDA 11.4.2, and cuDNN8. The Python container uses Python 3.8.10 and the following deep learning package versions: JAXlib 0.4.1, TensorFlow 2.11.0, torch 1.13.1, and MXNet 1.9.1. The Julia container uses Julia 1.8.5 and the following deep learning package versions: Flux 1.4.10, KNet 0.13.13, and CUDA 3.13.1.

III. Results

Figure 2 shows the loss and test accuracy metrics for the two models. For the MLP, the loss and accuracy metrics are comparable across frameworks. However, for the CNN, Flux has a much higher training loss and lower test accuracy. Although the goal is not to have the task performance metrics be identical, having the metrics being comparable is a good sanity check for ensuring the models are written as similarly as possible. One of the uncontrolled variables that might contribute to the differences in performance metrics is the parameter initialization strategy. For example, PyTorch uses Kaiming uniform as the default initialization in its Linear layer, while PyTorch uses Glorot uniform as the default initialization in its Dense layer.

Figure 3 shows the runtime metrics for the two models. For the MLP, PyTorch has the longest training time and inference time. The Julia frameworks have a longer total training time than TensorFlow and MXNet, but their average epoch training time was faster than that of the dynamic Python frameworks. This suggests that the Julia compile time improves the per-epoch performance, but overall is a penalty in this small model. Despite also including JIT compile time, JAX has the fastest total training time of 13.6 s and the fastest average epoch training time of just 0.94 s. However, JAX's inference time is slower than the Julia frameworks having an average batch inference time of 1.27 ms compared to under 0.6 ms for both Flux and KNet. However, the other frameworks show a considerable speed up in inference time compared to PyTorch, which takes nearly 20 ms to process a batch.

For the CNN, JAX is the fastest across all three runtime metrics, with a total training time of 108 s, average epoch training time of 3.72 s, and average batch inference time of 3.08 ms. Interestingly, the next fastest training time is PyTorch, which performs the worst in the MLP, although its average epoch training time is 8.57x slower than JAX. Although it is JIT compiled, Flux has an average epoch training time similar to that of TensorFlow, which was 17x slower than JAX. However, it was observed that during training, the GPU was underutilized during Flux training, suggesting that some of the CPU-bound data processing might be hindering its performance. Despite their slower training times, TensorFlow and Flux have considerable speedups during inference, especially

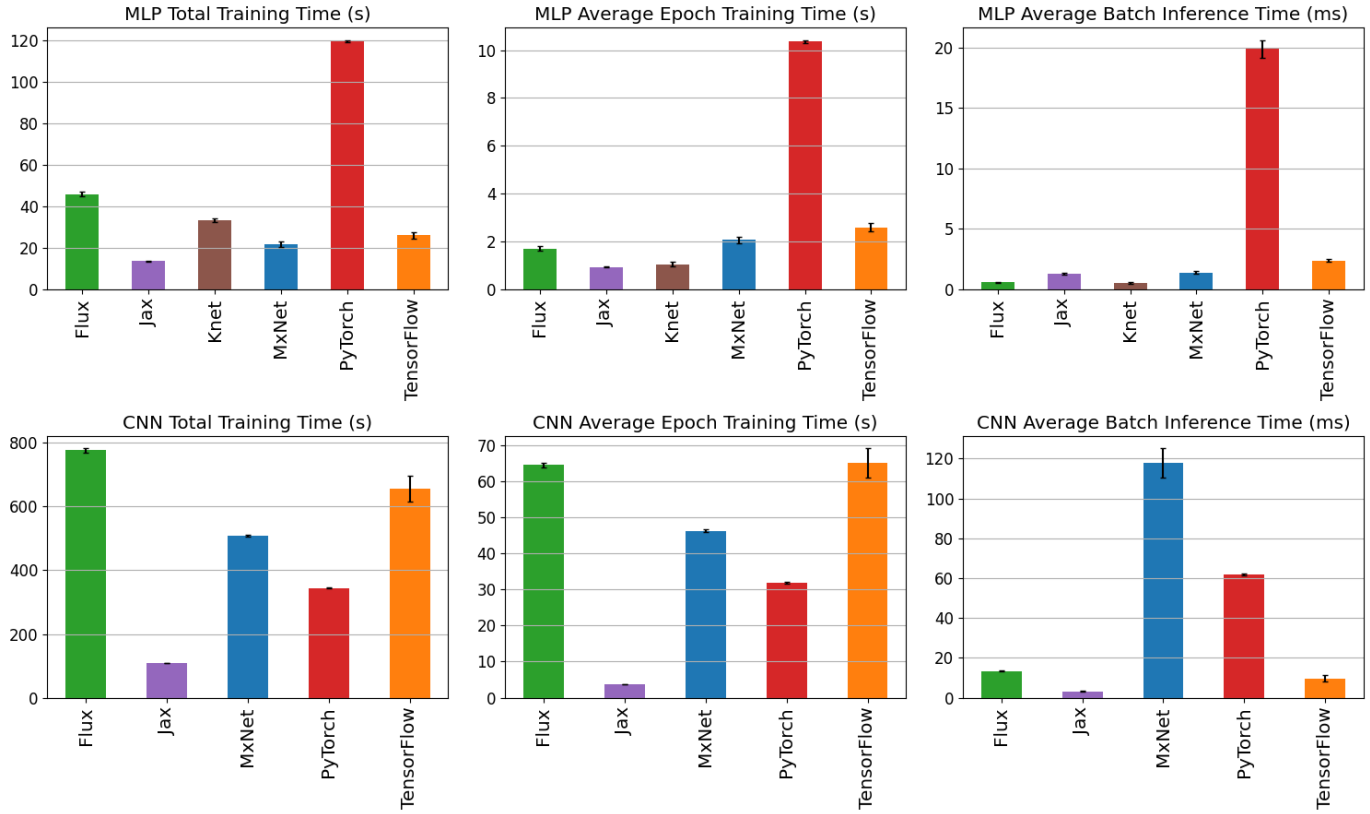

Fig. 3. Runtime metrics for MLP and CNN models (lower is better)

compared to MXNet and PyTorch.

A. Discussion

One advantage of JAX is that the XLA compiler is optimized for performing linear algebra operations as used in neural networks. In the case of Flux, the compilation and AD is performed on Julia code, which is not necessarily optimized for linear algebra. Perhaps due to residuals in the CNN model, it can be harder to run AD compilation in Julia, so Flux does not enjoy the same JIT compile speedups as it did in the MLP. Despite the JIT compile time, JAX was faster than the other three Python frameworks in all runtime metrics.

Although JAX is the fastest Python deep learning framework in these tests, it comes with some limitations that curtails its widespread use. The JAX API is much more difficult to use PyTorch and TensorFlow, particularly for writing code to setup training for the models. Rather than a consistent object-oriented API for training, a JAX developer will need to explicitly define the way the state of a model's parameters are applied to process input data and updated based on a loss function. There are also limitations to kinds of functions that can be JIT compiled with JAX, such as static array shapes, no in-place array updates, and writing pure functions. Poorly written JAX code will not result in performance gains and may not even run. Using neural network primitives in Flax helps, but not all primitives are implemented. Often, we want custom operations so model developers would need to be familiar with how to implement these in a JAX-compliant way to reap the benefits of its runtime efficiencies. Overall, using JAX for deep learning is much less developer-friendly than frameworks like TensorFlow and PyTorch.

IV. Conclusion

Through this work, we benchmark six deep learning frameworks in Python and Julia through building models from scratch for a MLP on the MNIST dataset and a ResNet on the CIFAR-10 dataset. Through running these benchmarks, we find that JAX is the fastest Python framework in both total training time and average batch inference time, despite the first metric including its JIT compile time. However, it is a challenge to use JAX effectively compared to other popular frameworks like PyTorch and TensorFlow.

In future work, we want to create and benchmark more models, such as generative image models like the variational autoencoder, and transformer models. These additional models would help give a sense of how each framework scales with the size and complexity of models. Additionally, with the portability of the Docker environment and open source code, we want to run these benchmarks across different hardware systems to verify whether the results hold.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[2] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.

[3] "Stack overflow developer survey 2022," 2022.

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems," *CoRR*, vol. abs/1512.01274, 2015.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[6] M. Innes, E. Saba, K. Fischer, D. Gandhi, M. C. Rudilosso, N. M. Joy, T. Karmali, A. Pal, and V. Shah, "Fashionable modelling with flux," *CoRR*, vol. abs/1811.01457, 2018.

[7] M. Innes, "Flux: Elegant machine learning with julia," *Journal of Open Source Software*, 2018.

[8] D. Yuret, "Knet: beginning deep learning with 100 lines of julia," in *Machine Learning Systems Workshop at NIPS*, vol. 2016, p. 5, 2016.

[9] C. C. Margossian, "A review of automatic differentiation and its efficient implementation," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, p. e1305, 2019.

[10] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[11] M. Innes, "Don't unroll adjoint: Differentiating ssa-form programs," *CoRR*, vol. abs/1810.07951, 2018.

[12] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, "Flax: A neural network library and ecosystem for JAX," 2023.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016.

[14] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

Classifying Motor Imagery from Open Source Datasets

Liam Schultz
Western University
lschult7@uwo.ca

Pratik Gupta
Western University
pgupta85@uwo.ca

Srinjoy Choudhury
Western University
schoud29@uwo.ca

***Abstract*—The goal of this paper was to start learning about how BCI technology works and specifically how to use an EEG headset to predict motor imagery. The method that was attempted was to use the CSP method along with a Stockwell transform to process the data. A CNN was then used as the model to try to predict when a subject was thinking about performing movements with their left or right hand. The accuracy achieved was only 60% on the validation data which is much lower than expected. Hopefully this project will continue to be worked on to improve the accuracy and gain further knowledge in this emerging subject area.**

I. Introduction

A. Motivation

Brain-Computer Interface (BCI) technology is an exciting emerging field with applications from aiding those with disabilities to gaming. One of the most widely known groups developing this technology is Neuralink. Their method of getting data is by implanting electrodes in the brain [1]. However, this is an invasive medical procedure with numerous ethical concerns. Instead of that, the proposed approach was to use a traditional electroencephalogram (EEG) device to collect data and then predict the movement of a user's limbs. This is not a particularly groundbreaking idea, however processing EEG data using AI is a complex subject and this paper represents a first attempt at understanding and applying some common methodologies. Hopefully this paper will serve as a guide to someone interested in the field and eventually a marker of the progress that has been made.

B. Related Works

One resource that was found was a paper from 2022 by Chacon-Murguia and Rivas-Posada covering the classification of motor imagery (where the subject imagines performing a movement without actually performing any movement) using a Convolutional Neural Network (CNN) and a variety of processing methods. The accuracy they managed to achieve was between 86.47% and 97.67% depending on the processing methods used. [2] The system they propose is fast enough to classify 4 different classes of motor imagery in real time. CNNs are a common type of model used in this field and with the variety of methods used it was decided that a good first step would be to replicate this study.

C. Problem Definition

Originally the plan was to get access to an EEG device to generate data and then use that data to train and validate various methods. However, due to budget constrains access to an EEG device could not be obtained. Instead, publicly available datasets were used to prove the concept. Some early experimentation was done with the grasp-and-lift dataset [3] (Note: the dataset is currently hosted on kaggle), but the BCI-IV-2a dataset [4] was eventually settled on because it is clearly labelled and also used in many papers. This allowed for simpler comparisons to the results obtained in other papers.

The most accurate method from [2] was used. Which was to use Common Spatial Patterns (CSPs) and a Stockwell transform on the data and then use a CNN for classification.

II. Methodology

A. Data

The data obtained from the BCI-IV-2a dataset is composed of 22 channels of EEG data and 3 channels of electrooculogram (EOG) data stored in a .gdf file (General Data Format for Biomedical Signals) [4]. To generate the data 9 subjects were asked to imagine moving their left and right hands 72 times each, yielding 144 trials per subject. Note that the dataset includes two more classes of motor imagery but they were ignored to simplify the problem. Check the above citation for a full description. The method used only included 8 of the EEG channels. The MNE library handled many of the complex calculations needed and generated the visualizations [5].

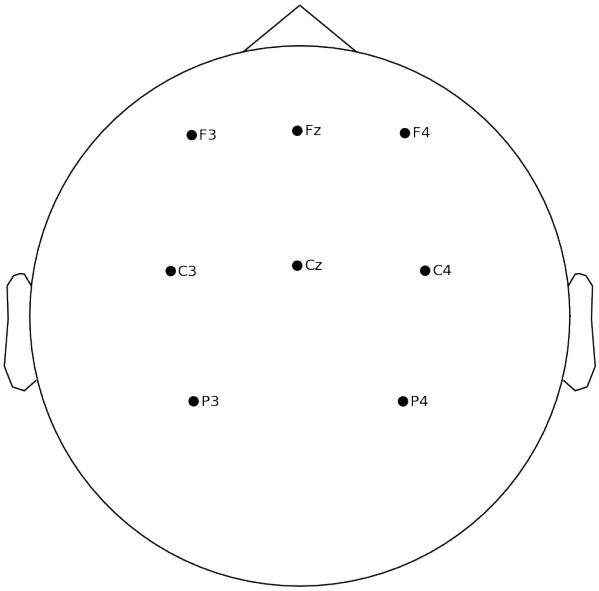

Fig. 1. A montage depicting the sensors used.

Figure 1 shows the which sensors were used and where they are located on the head. The labels correspond to the 10-20 system.

B. Filtering

A 7.5 to 30 Hz 4th order Butterworth band-pass filter was used to filter each channel because the alpha and beta waves (the brain waves in the 7.5 to 12 and 12 to 30 Hz ranges) are the ones relevant to motor imagery [2]. The following figures show the data before and after being filtered with dashed vertical lines representing 7.5 and 30 Hz.

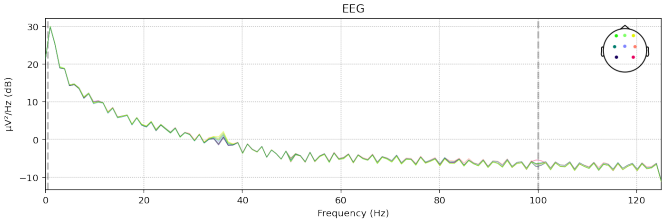

Fig. 2. A channel before being filtered.

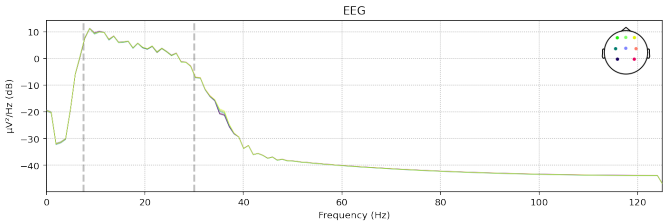

Fig. 3. A channel after being filtered.

C. Epoch extraction

From each trial in the dataset 2 seconds of activity were captured and labelled according to which action they were performing. At this point the trials involving feet and tongue motor imagery were discarded. The trial labels were also extracted and encoded into a one-hot encoding.

D. CSPs

CSPs use linear algebra to compute a matrix from the eigenvalues of the matrix of channels of the labelled data. Multiplying new data by this matrix will project it into a subspace of lower dimension. In this case it goes from having 8 channels to having just 2. The matrix will be unique to the subject but once the data from all the subjects is multiplied by the its subject's matrix it can all be used to train the CNN. The exact mathematical method is detailed in other papers which also give a better understanding of how this method works [6]. In the method used the CSP matrix was generated using the training data and then applied the matrix to the testing data

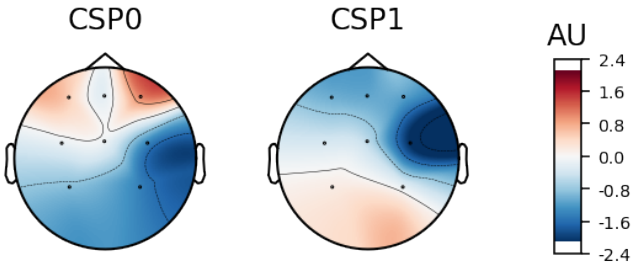

Fig. 4. A topographical representation of the CSP filter.

Figure 4 shows what regions of the brain tend to correlate with motor imagery. The red regions indicate that there is some correlation (whether positive or negative) where the blue regions indicate that there is typically no correlation. The units of the scale are arbitrary.

E. Stockwell transform

The Stockwell transform is a method of transforming the data from the time domain to the frequency domain. It is similar in function to a Fourier transform but the method is different.

Bicubic Interpolation

Depending on the parameters set for the CSPs and Stockwell transform the data may be either too large or in an awkward shape. Ideally the desired image is one that pooling can be applied to without shrinking it beyond the size of the kernel. Likewise, too large of an image would be require more memory and more computation time. So a bicubic interpolation was applied, which is a common method of shrinking images, to make the output into a 22x100 matrix.

F. CNN

The CNN used 3 convolutional layers of size 128 with 3x3 kernels and ReLU activations. After each convolutional layer there is a max pooling layer with a 2x2 pool size and a stride length of 1. After all of these layers the results are flattened into a fully-connected layer which feeds into a layer of 2 neurons with a softmax activation which represent the left and right hands.

III. Results

The model was only able to achieve an accuracy of approximately 60% on the validation data. Unfortunately it overfits to the training data quite rapidly even when dropout layers are used. This is far below the expected performance. Reviewing the paper [2] the expected accuracy with only the CSP method is over 80%. This indicates that the CSP method used is likely flawed. Migrating away from the MNE library is a potential solution. However, the number of epochs extracted was also lower than the number extracted in the paper. For each epoch extracted they extracted 8 overlapping epochs which provided much more training data. MNE does not have an easy way of doing this so a custom epoch extraction method will likely have to be written.

IV. Conclusion

In this project some of the basic techniques that are used to process raw EEG data into data which is more comprehensible to a CNN were covered. Typically the process is to do filtering on the data, pass it through the CSP method and then do some form of transform from the time domain into the frequency domain. This paper used the Stockwell transform but the Continuous Wavelet Transform (CWT) or even a Short-Time Fourier Transform (STFT) could also work. The next steps are to try to get somewhere around an 80% accuracy using only the CSP method. Once that is achieved a transform can be applied to improve the accuracy further. Another option is do more research to find other sources and try to implement some of the techniques employed in them. Additionally, obtaining an EEG headset to record data is a goal for the future of the project. Once a grasp on the common methods used in this field has been attained more exploratory research could be done.

References

[1] Approach. Neuralink. (n.d.). Retrieved March 12, 2023, from https://neuralink.com/approach/

[2] Chacon-Murguia, M.I., Rivas-Posada, E. A CNN-based modular classification scheme for motor imagery using a novel EEG sampling protocol suitable for IoT healthcare systems. Neural Computing for IOT based Intelligent Healthcare Systems (2022). https://doi.org/10.1007/s00521-021-06716-x

[3] Luciw MD, Jarocka E, Edin BB (2014) Multi-channel EEG recordings during 3,936 grasp and lift trials with varying weight and friction. Scientific Data 1:140047. www.nature.com/articles/sdata201447

[4] Brunner, C., Leeb, R., Müller-Putz, G. R., Schlögl, A., Pfurtscheller, G. (2008). BCI competition 2008 - Graz data set A. berlin brain-computer interface (BBCI). Retrieved March 12, 2023, from https://www.bbci.de/competition/iv/desc_2a.pdf

[5] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. Frontiers in Neuroscience, 7(267):1–13, 2013. doi:10.3389/fnins.2013.00267.

[6] Wang, Y., Gao, S., & Gao, X. (2005). Common Spatial Pattern Method for Channel Selection in Motor Imagery Based Brain-computer Interface. In 2005 27th Annual International Conference of the IEEE engineering in Medicine and Biology Society (pp. 5392–5395). Shanghai, China; IEEE.

Comparing AI Navigation Methods Using Counter-Strike: Global Offensive

Michael Salton
Western University
msalton@uwo.ca

Ethan Pisani
Western University
episani2@uwo.ca

Swayam Sachdeva
Western University
ssachd29@uwo.ca

***Abstract*—This paper describes two AI agents built to play the first-person shooter video game "Counter-Strike: Global Offensive" (CS:GO). The first agent uses a combination of YOLOv7 object detection and A* pathfinding algorithm. The second agent uses a deep neural network trained on a large data set of video footage of professional CS:GO matches using behavioural cloning and offline reinforcement learning. While most "aim-bot" programs tap into the games memory to retrieve information such as the coordinates of players and aim the mouse directly at them, the goal for our project was to create an AI that can match the performance and behaviour of real players while relying purely on the visuals on the screen. We look to compare the behaviour and realism of an AI built with pathfinding vs an AI built with machine learning.**

I. Introduction

Counter-Strike: Global Offensive is a popular first-person shooter video game that has gained immense popularity among gamers worldwide. With the rise of artificial intelligence (AI), there has been increasing interest in developing AI-based systems capable of playing complex games such as CS:GO. In recent years, researchers have focused on developing AI bots that can compete against human players in online matches. These AI bots are designed to learn from the gameplay data, enhance their strategies, and eventually surpass human players in their gameplay performance. However, these projects mostly focused on games with convenient APIs that make it easy to scrap data from the game. Whereas for CS:GO, these tools are not available, which precludes us from using these common techniques.

In this research paper, we present a novel AI-based system that has been developed to play CS:GO. The AI agent is designed to learn and adapt to different game scenarios and player styles. Our research aims to explore the capabilities of machine learning approaches to problems and compare that to the typical pathfinding-like AI, in both video games and real-world scenarios. The paper provides insights into the development of machine learning based systems for gaming applications and the challenges faced in designing effective AI bots for video games and real-world systems.

A. Motivation

The use of behavioural cloning and reinforcement learning presents a promising approach to creating an AI that can effectively play CS:GO. Behavioural cloning, which involves training an agent to mimic the actions of expert players, provides a good starting point for building an intelligent agent. Reinforcement learning, on the other hand, enables the agent to learn from its own experiences in the game environment and improve its decision-making over time.

The ultimate goal of this research is to develop an AI agent that can compete with human players in CS:GO. Such an agent would have numerous practical applications, including enhancing the game experience for players by making AI in video games more realistic and potentially even applying these techniques to real-world scenarios involving AI.

We aim to distinguish between "dumb" and "smart" AI and explore the advantages and disadvantages of each. We define "dumb" AI as relying on traditional pathfinding techniques, where movements are predetermined or precalculated by the computer. This type of AI has been commonly used in video games since the inception of the industry, including popular titles like Mario, Half-Life, Elder Scrolls, and Grand Theft Auto. On the other hand, we define "smart" AI as utilizing dynamic, real-time calculations, that is built with some form of machine learning approach. We will discuss the two AI agents we have built, one "dumb" and one "smart", and see how they compare to each other. What kinds of advantages does either AI bring and what are the problems associated with each technique. Finally, thinking about the future of video games, what kind of AI will be the better option for taking the gaming experience to the next level. Our paper mainly focuses on the navigation aspect of AI. The reason we chose to use CS:GO as our test field as opposed to an environment that is based purely around navigation, like a maze for example, is because; generally in video games, the AI has other actions it performs alongside movement and we wanted to see how these actions affect the AI's movement and what steps need to be taken to keep them from "breaking".

This research will not only contribute to the field of AI by advancing our understanding of how to build intelligent agents that can play complex games like CS:GO, but it will also have practical implications for the gaming industry and beyond. We believe that the results of this research will be of significant interest to game developers, players, and researchers alike, and will pave the way for further research into the interesting overlap of AI and video games.

B. Related Works

We came across a paper on behavioural cloning for CS:GO. Entitled: "Counter-Strike Deathmatch with Large-Scale Behavioural Cloning", the paper goes over how Tim Pearce and Jun Zhu designed a convolutional neural network (CNN) and a long short-term memory (LSTM) neural network to play CS:GO only training on video frames and user input. This is the basis for the "smart" AI model we envisioned. They used 93 hours of online messy data along with a few hours of expert data to train the model. It resulted in a medium-level CS:GO bot in difficulty. The model learned to navigate the 3d environment and was able to identify and shoot enemy players. It also learned human-like strategies like spray control while also having its own strategies. It can still be beaten by a human more times than not, but it is very promising research into the aspect of learning strategy from watching players play the game.

C. Problem Definition

A simple path-finding-like approach may be an easy and common solution for many of today's technologies such as video games, exploration, and industrial automation, to name a few. With video games, probably the most obvious example, pathfinding algorithms have been in place since the beginning. Most games work by having the computer direct characters around the map via a predefined path, or a set of predefined locations. The developers can set up more intricate characteristics to these paths such as a set of rules stating certain paths are preferable over others as some may contain some kind of "risk" or "reward", as well as dealing with dynamic in-game objects. Much of the time these dynamic in-game objects are also under the computer's control, so the program may have to account for a multitude of different entities and move them all in accordance with each other.

Multi-agent path finding (MAPF) is a pathfinding system that has two or more agents navigating the graph. There is a set of common problems with MAPF which are, vertex conflict (two agents attempt to navigate to the same vertex), edge conflict (two agents attempt to use the same edge in the same direction), swap conflict (two agents attempt to use the same edge but in opposite directions (swap positions)), and follow conflict (when one agent attempts to occupy a position at a given time h that was occupied by another agent at time $h-1$). These conflicts can be hard to deal with and have the potential of causing problems within many different applications of a pathfinding system if not properly handled. In the case of the A* algorithm, a solution for these problems is to move all possible agents from one vertex to a free neighbouring vertex one at a time and then initiate the pathfinding process again. This results in a search space of $|V|^K$, where V is the number of vertices on the map and K, is the number of agents, and a branching factor of $\frac{|E|}{|V|}^K$, where E is the total number of possible outgoing edges that an agent can take. These exponential complexities make the A* algorithm intractable for large-scale multi-agent pathfinding problems, as the search space and branching factor quickly become too large to search efficiently.

Not only can these issues have impacts on the game's performance but it makes for an unrealistic and non-immersive gaming experience. AI in many video games today is known to be rather dumb as they don't react in realistic ways to their environment. A famous example of this is Bethesda's "The Elder Scrolls V: Skyrim". The AI in this game is notoriously unrealistic. Non-player characters (NPCs) in Skyrim often walk into one another, randomly stop following you when they're meant to follow you, and overall have quite non-human-like behaviour. Many players describe poor AI as "a reminder it's just a game", which is not what you want when you are trying to get immersed into a new world. Machine learning approaches may offer a better solution for this problem and that is exactly what we set out to learn with this research.

II. Methodology

The design of our two AI agents was done sequentially, starting with the "dumb" one. We gave this agent the name "Atlas". The first set of data was a collection of screenshots from various CS:GO games, personally collected and labelled by our team. We used this data to train our YOLOv7 object detection model. We used OBS to screen record hours of gameplay and then FFmpeg to convert this footage into individual frames, and Roboflow to label them. In total, our dataset was compromised of about 3000 images. We ensured at least 75% of the images contained people, the remaining were blanks. We labelled all the people found in the images with bounding boxes and applied some augmentations to the data such as blurs, shears, and brightness changes to represent some in-game occasions like flash and smoke grenades.

TABLE I
Information on the number of images used for the YOLOv7 model.

Type	Training	Validation	Testing	Total
Terrorists (T)	1268	178	7	1453
Counter-Terrorists (CT)	1343	191	7	1541
Blanks	137	25	3	165
All	2748	394	17	3159

A. Object Detection

With data collection complete, we used transfer learning to modify the YOLOv7 model. We trained a model with some augmentation then we increased the model size and image resolution input and refined the augmentations. Our best model ends with 87% accuracy at 0.5map. We then converted the model weights to run using NVIDIA Tensor-RT for inference and got the model to shoot at the detected players. We made a function to move the mouse to the middle of each detected bounded box, 2/3 of the way to the top using pynput. With this done Atlas was able to detect, and shoot enemies in the game.

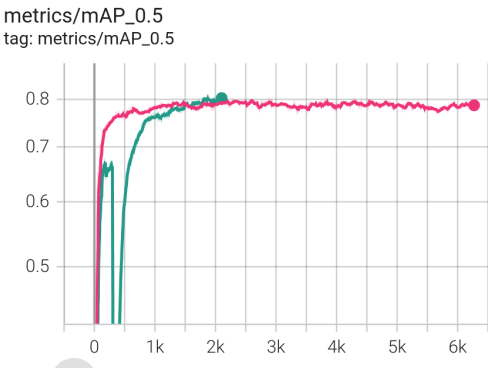

Fig. 1. This map@0.5 graph displays two versions of a the YOLOv7 model, with v1 represented in pink and the v2 represented in teal, where the highest value for v1, 0.78 and 0.81 for v2.

B. Movement

Our second dataset for Atlas was a list of in-game coordinates from around the map representing vertices to be used by the A* algorithm. We went through many different iterations of this graph before we finally landed on one that worked. It was difficult because we had to ensure that the AI could get to any location on the map by strictly walking along the edges between each vertex, as well as ensuring that if two vertices are connected it is possible to walk between them in the game (there are no obstacles obstructing that path). We used this along with a bit of code for sending inputs to the game using the pynput library. The A* algorithm takes a start point and an endpoint on a matrix of 1s representing vertices and 0s representing obstacles with the same cardinality as the waypoint matrix. It then finds the shortest path along the edges from the start point to the endpoint.

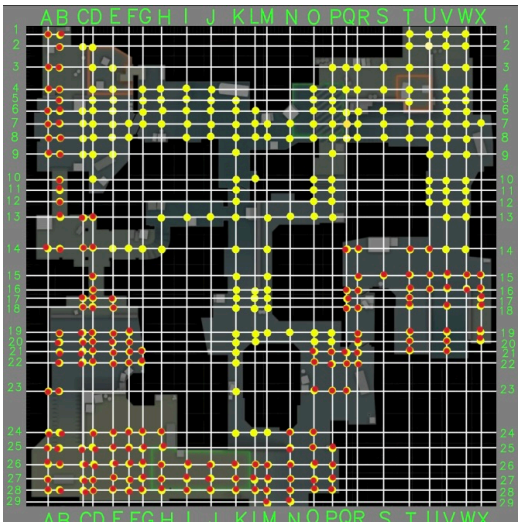

Fig. 2. Graph of vertices used in the A* algorithm.

Unfortunately, the player's current coordinate was not something that was available with CS:GO's game state integration so we were unable to use the in-game coordinates we had collected. Therefore, we had to purely rely on pressing "W", "A", "S", and "D" for a certain length of time, based on the distance between vertices and hope that it would line up perfectly. This caused many issues of its own because if the agent hit any kind of obstacle, like another player, it would be off course and have to be repositioned.

Fig. 3. A* algorithm running on a simulated "Dust II" map.

C. Reinforcement Learning

To optimize the performance of our reinforcement learning agent, named "P-body", we employed a reward function during training. The reward function was defined as follows: $reward = s_8 - 0.5s_{10} - 0.02\sum_{i=1}^{N} s_i$, where s_8 and s_{10} represent specific states and $\sum_{i=1}^{N} s_i$ represents the sum of all states. Our network was configured to accept input dimensions of (200,200,3) for screen pixels, (58) for states, and (62) for output confidence for action space. We utilized behavioural cloning and OpenAI Gym's CS:GO environment to train P-body through offline reinforcement learning. We sourced our training data from past research that involved 93 hours of scraped online data and professional match data containing CS:GO gameplay, presented as a sequence of frames accompanied by player metadata and user input. Offline reinforcement learning was chosen due to its capacity for batch training on powerful hardware, enabling training at a speed that surpasses real-time. JAX/FLAX was used for its exceptional parallelization capabilities and optimized JIT compiler, resulting in improved machine learning training speed. The AI was designed to predict confidence values for user inputs including movement, mouse control, and weapon swapping using the BCLeaner. The incorporation of reinforcement learning allows the AI to learn and improve its performance over time through a reward optimization process, where it receives rewards for

desirable actions and penalties for undesirable ones. Our AI was rewarded for killing enemies during gameplay.

Fig. 4. Graph illustrating the change in loss value for a reinforcement learning model over the course of its training with an end in 58.206.

III. Results

A. Atlas

Atlas would always run into other players which would cause him to be off course from the grid and have to be repositioned. We ran two instances of the AI in the same game, to test multi-agent pathfinding, and the results were like we expected. The two agents would often succumb to vertex conflict, edge conflict, and swap conflict, resulting in them being off-grid and having to be repositioned. Overall, the movement of Atlas seemed very robotic, he often would pause while the next instance of the algorithm is being calculated, and running into obstacles was a common occurrence for him. This type of robotic movement is not something that players want to encounter when playing against or with this type of AI. It makes the game feel fake, reducing players' engagement. Atlas was comparable to the easy built-in CS:GO bots.

B. P-body

On the other hand, P-body was not quite as good at shooting as Atlas was with the YOLOv7 model, but he was much better at moving, and that is really what we are testing. With more training, P-body would be able to become much more proficient at shooting, and eventually overcome Atlas, so that is not an issue. Since P-body did not run using pathfinding the problems with MAPF that Atlas was encountering were not an issue. P-body was smart and was making decisions in real-time, because of this, his movements seemed much more natural compared to Atlas. P-body would never run into another player, obstacle, or wall, he would not make random stops nearly as often as Atlas did, and overall he seemed a lot less robotic and more human-like than Atlas. P-body was comparable to the hard built-in CS:GO bots.

IV. Conclusion

All in all, our project accomplished two AI agents that can traverse an area of space. We set out to determine if pathfinding is an adequate approach to these types of problems or if vying for an alternative approach is worth the time. Atlas the "dumb" AI is compromised of an object detection model used for detecting enemies in the game, and a pathfinding algorithm to move around. P-body the "smart" AI was built using behavioural cloning and offline reinforcement learning, with the aim of cloning the behaviour of professional CS:GO players.

A. "Dumb" vs "Smart"

We determined that using a machine learning approach to AI movement can reduce many conflicts that are present when using a traditional pathfinding approach. Pathfinding has been a common technique in video games and real-world applications for a very long time and perhaps it's time companies start taking advantage of these more modern techniques of machine learning, specifically reinforcement learning. We found that even with the small amount of training our reinforcement learning model had, it was still better at navigating than the pathfinding model. P-body did take more time, knowledge, and resources to construct. It took days of training, data collection and manipulation, programming, and research to be able to get him to function properly. Whereas for Atlas, the amount of knowledge needed to construct him was a lot less. Less time and resources were needed, not to mention there are decades of previous AIs that use the same method. All considered, constructing AI with the "smart" AI approach does produce much higher quality and desirable results.

B. Future

Moving forward, we would like to train P-body even more, and truly see what behavioural cloning is capable of doing. We would like to see him pick up on the "unspoken rules" of CS:GO and see just how close he can mimic the behaviour of humans. The ability to pick up on unspoken rules in video games is particularly important for realistic and immersive gaming experiences. In games like CS:GO, there are often unwritten rules and conventions that players follow, such as how to move around the map, when to engage in combat, and how to coordinate with teammates. If an AI could pick up on these rules and replicate them, it could make the game feel more natural and lifelike. The future of gaming depends on the research being done today and AI is at the front of it. Realistic AI in video games has been something companies have been trying to nail for years, perhaps a machine learning approach will become the mainstream of future video game AI.

References

[1] I. Kostrikov, "JAXRL: Implementations of Reinforcement Learning algorithms in JAX," GitHub, Oct. 2022. [Online]. Available: https://github.com/ikostrikov/jaxrl2. [Accessed: Mar. 13, 2023].

[2] T. Pearce and J. Zhu, "Counter-Strike Deathmatch with Large-Scale Behavioural Cloning," arXiv:2104.04258, 2021.

[3] Pouke, M.: Using GPS data to control an agent in a realistic 3D environment, pp. 87–92. IEEE, September 2013. https://doi.org/10.1109/NGMAST.2013.24

[4] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," arXiv preprint arXiv:2207.02696, 2022.

[5] C.-Y. Wang, H.-Y. M. Liao, and I.-H. Yeh, "Designing Network Design Strategies Through Gradient Path Analysis," arXiv preprint arXiv:2211.04800, 2022.

Our GitHub Repository: https://github.com/michaelsalton/ProjectLambda

DCL Crowd Counting

Benjamin Hui
Queen's University
18bh13@queensu.ca

Jan Karcz
Queen's University
jan.karcz@queensu.ca

Bartek Kowalski
Queen's University
21bmk9@queensu.ca

Sebastian Deluca
Queen's University
20sad4@queensu.ca

Jordan Corbett
Queen's University
21jjkc@queensu.ca

***Abstract*—This paper proposes a solution to the problem of long queues at ATM's in Kenya, using the YOLOv8 algorithm to count the number of people waiting in line outside an ATM. The paper discusses the motivation for crowd counting using artificial intelligence techniques and reviews related works in the field. The authors define the problem of long queues at ATMs in Kenya and explain how this problem can hinder economic growth and productivity. The proposed solution aims to reduce waiting times, improve the customer experience, and increase the efficiency of ATM usage by providing customers with information about the shortest queue. The methodology involves using the ShanghaiTech dataset and iterative refinements to enhance the model's performance and incorporates advanced techniques.**

I. Introduction

The accessibility and availability of financial services, particularly the Automated Teller Machines (ATMs), are critical to the smooth operation of economies worldwide. However, long queues of people waiting to withdraw cash from ATMs are commonplace in many parts of the world, particularly in developing countries. Kenya, a developing country in East Africa, is no exception. Long queues are a regular occurrence outside ATMs, particularly in urban areas, with customers sometimes waiting for hours to access their cash. The long waiting times can be frustrating, leading to unproductive hours and reducing the quality of life for customers. Furthermore, in a country where cash is still the primary mode of payment, the ATM queues also lead to a reduced rate of transactions, hindering economic growth.

To tackle this problem, we propose a solution that uses the YOLOv8 algorithm to count the number of people waiting in line outside an ATM. This solution can provide valuable information to the customers, allowing them to choose the shortest queue and thereby reducing their waiting time. YOLOv8 is an object detection algorithm that can detect and count multiple objects in an image with high accuracy. In this research, we aim to implement YOLOv8 to count the number of people in line at ATMs in Kenya and use this data to provide the shortest queue to customers, thereby reducing their waiting time and improving their overall experience.

A. Motivation

With many emerging applications of crowd counting, and the collection of data around the globe; crowd counting has an increasingly important role in modern daily life. From counting the line at an ATM, to detecting crowd surges at the most recent world cup [1] the data gained from crowd counting benefits society [2]. Over the past years, everyone has been told to stay 6 feet away from another, but with a large population, enforcement of such is near impossible. With the use of artificial intelligence techniques, these measures may be enforced to keep everyone safe. This was done in Saudi Arabia using Deep Convolutional Neural Network (DCNN) [3]. This gave two valuable insights while trying to control the pandemic; the ability to detect crowd formations, and calculating the population density in specific areas. This system allowed for pedestrians to be notified when entering an unsafe area, keeping everyone safe. Moreover, the same technology may be used to inform citizens of busy locations with long wait lines, to allow for higher efficiency while performing daily tasks. Specifically, the technology can inform citizens of Kenya of long lines at ATMs, allowing for long wait times to be avoided.

B. Related Works

Zhang et al. proposed a multi-column convolutional neural network (CNN) in 2016 for crowd counting and used multiple columns to learn different features from an input image. The density maps generated by each column were combined to obtain the output count. The weights of the combinations of the weighted sum were adjusted during training and was able to capture both global and local features effectively.[4]

In 2018, Sam et al. proposed a method for crowd counting using synthetic data in the wild. Annotated data was difficult to come by in a real-world setting and proposed to generate synthetic data using a combination of 3D modeling and rendering techniques. Using a deep learning approach they trained a CNN to process data that has been augmented with both the synthetic and real-world data. This is done by overlaying synthetic crowds onto real-world scenes or backgrounds. The crowds are also adjusted by enlarging or adding noise to the images. Through evaluating their performance on different datasets, their methods resulted in a significant improvement over previous methods from its time.

C. Problem Definition

The accessibility ATM's is a crucial aspect of daily life in Kenya. However, long queues outside ATMs have become a persistent problem in urban areas, with customers often waiting for hours to access their cash. The uneven distribution of queues is another critical problem, with some ATMs having long lines while others are relatively free. This issue creates an inefficient distribution of resources, with people unnecessarily waiting in long lines while other ATMs remain underutilised.

The problem of long queues at ATMs can lead to a reduction in the rate of transactions, hindering economic growth and productivity. The purpose of this research is to address this very problem. We propose using the YOLOv8 algorithm to count the number of people waiting in line outside an ATM and provide customers with information about the shortest queue. By using this data to direct customers to the shortest queue, we aim to reduce waiting times, improve the customer experience, and increase the efficiency of ATM usage.

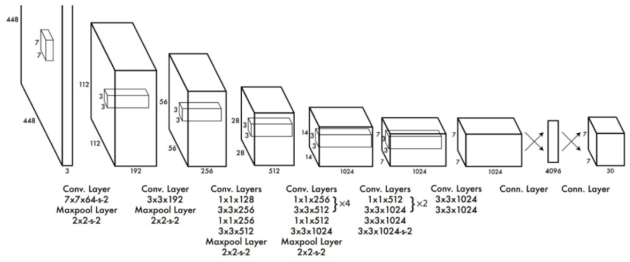

Fig. 1. YOLOv8's architecture

II. Methodology

Our goal is to continuously improve the model's effectiveness by employing innovative methodologies and leveraging cutting-edge technologies to boost its precision and reliability. By taking the YOLOv8 model we planned to use the ShanghaiTech dataset to enhance the model's performance by implementing iterative refinements and incorporating advanced techniques to optimize its accuracy.

1) The ShanghaiTech dataset is a benchmark dataset for crowd counting in computer vision with 1,198 images with different ranges of lighting and motion. Although the dataset does produce a bias towards Asian people, it consists of a diverse range of scenes that makes it a valuable resource for training models.
2) Our proposed solution is to enhance the performance of the YOLOv8 model by implementing iterative refinements and incorporating advanced techniques to optimize its accuracy on the ShanghaiTech dataset. Specifically, we use data augmentation techniques, such as random cropping, flipping, and rotation, to increase the diversity of the training set. We also plan to incorporate attention mechanisms, such as spatial and channel attention modules, to help the model focus on important features and improve its accuracy. Additionally, we will experiment with different loss functions, such as the mean absolute error (MAE) and the mean square error (MSE), to find the best combination of parameters that yields the highest accuracy.

III. Results

Our proposed solution, which involved iterative refinements and advanced techniques to optimize the model's accuracy, was tested on the ShanghaiTech dataset, where it achieved an impressive accuracy rate of 99% in evaluating crowds of people in urban settings. This high level of accuracy indicates that our solution is highly effective and reliable in such scenarios.

Our proposed solution has significant potential for use in a variety of applications, including public safety, transportation planning, and urban design. Its accuracy in evaluating crowds of people in urban settings can help inform decisions related to these fields, and its use could lead to more effective and efficient planning and decision-making.

However, it is worth noting that our model does have limitations. Specifically, its performance decreases when evaluating pictures with poor lighting or when people are further off into the distance. In such scenarios, the accuracy rate drops to 58%, indicating that the model is less effective at identifying smaller objects. This limitation is to be expected, as smaller objects are naturally more difficult to identify.

IV. Conclusion

In conclusion, this research proposes a solution to the problem of long queues at ATMs in Kenya, which can cause inconvenience and reduce the quality of life for customers while also hindering economic growth. The proposed solution involves using the YOLOv8 algorithm to count the number of people waiting in line outside an ATM, which aims to reduce waiting times, improve the customer experience, and increase the efficiency of ATM usage. The paper discusses the motivation for crowd counting using artificial intelligence techniques and reviews related works in the field. Our methodology involved selecting an appropriate crowd counting dataset, training the YOLOv8 object detection algorithm, developing a system to process the data and provide customers with information, and analyzing the results of our model when we used it in the real world. Through the use of the ShanghaiTech dataset and iterative refinements to enhance the model's performance, the proposed solution can provide customers with information about the shortest queue, thereby reducing their waiting time and improving their overall experience. Our results showed that the proposed solution is effective in counting the amount of people in an image, and can be used to shorten the overall wait time of people looking to make cash withdrawals in places where lines are often very long. Furthermore, the solution we proposed can be applied not only to ATMs but also to other similar financial services that often suffer from long waiting times. Future research could explore the scalability of the proposed solution and the potential for integrating it with other financial services.

Overall, this research provides a valuable contribution to the use of artificial intelligence techniques in solving real-world problems and can benefit society by enhancing efficiency and productivity. The proposed solution has the potential to significantly reduce waiting times, thereby improving the customer experience, and increasing the efficiency of ATM usage. Therefore, the solution can help improve the quality of life for customers and contribute to economic growth in Kenya.

Our research demonstrates the potential of using computer vision algorithms such as YOLOv8 to improve the efficiency and effectiveness of financial services such as ATM usage. In conclusion, our research provides a promising solution to the problem of long queues at ATMs in Kenya and contributes to the growing body of literature on the use of computer vision in improving financial services.

REFERENCES

[1] Elharrouss, Omar & Almaadeed, Noor & Abualsaud, Khalid & Al-ma'adeed, Somaya & Al-Ali, Ali & Mohamed, Amr. (2022). FSC-Set: Counting, Localization of Football Supporters Crowd in the Stadiums. IEEE Access. 10. 1-1. 10.1109/ACCESS.2022.3144607.

[2] Wang M, Cai H, Zhou J, Gong M. Interlayer and intralayer scale aggregation for scale-invariant crowd counting. Neurocomputing. 2021 Jun 21. doi: 10.1016/j.neucom.2021.01.112.

[3] Kammoun jarraya, Salma & Alotibi, Maha & Ali, Manar. (2021). A Deep-CNN Crowd Counting Model for Enforcing Social Distancing during COVID19 Pandemic: Application to Saudi Arabia's Public Places. Computers, Materials & Continua. 66. 1315-1328. 10.32604/cmc.2020.013522.

[4] GUO, Boqiu & JIN, Hua & CAI, Yong & WU, Lunpeng & SUN, Xianli & CHEN, Fan & CHENG, Ziyu. (2022). Application of object detection algorithm based on deep learning in classification of wild ginseng grades. 10.21203/rs.3.rs-2234086/v1.

EEG Brain-Computer Interface

Ethan Callanan
Queen's University
e.callanan@queensu.ca

Daniel Martin
Queen's University
martin.daniel@queensu.ca

Michael Barrack
Queen's University
18medb@queensu.ca

Kiarash Mirkamandari
Queen's University
21km83@queensu.ca

Jack Macaulay
Queen's University
john.macaulay@queensu.ca

***Abstract*—Brain-computer interfaces are devices that enable direct communication between the brain and a computer, allowing users to control various applications with their brain activity. An electroencephalogram, a device that can measure brain activity through electrodes attached to the scalp, can be used to build a non-invasive (not requiring surgery) brain-computer interface. This technology has a large potential impact for accessibility devices for those with motor impairments and for immersive gaming experience. The goal of this project was to build a non-invasive brain-computer interface binary game controller. To achieve this, we built and tested multiple models, including a statistical classifier, a convolutional neural network, and a singular vector machine. The statistical classifier achieved 81.36% accuracy, the neural network had 92.84% but struggled to generalize, and the singular vector machine achieved 94.43%. Finally, the models were integrated as the control mechanism for a custom version of the game Flappy Bird.**

I. Introduction

A. Motivation

Brain-computer interfaces (BCIs) are devices that facilitate direct communication between the brain and a computer, enabling the control of computer inputs (mouse or keyboard inputs), robots, or prosthetics. Neurotechnology and BCIs have gained increasing attention in recent years, jointly due to the popular public demos of Neuralink [1] and the ever decreasing cost of entry for non-invasive research. While invasive BCIs promise much higher acuity and wider applications, non-invasive BCIs such as those based on the electroencephalogram (EEG) can provide a reliable and affordable way to measure brain activity build BCI systems. EEG BCIs are especially attractive for building consumer technology, as they are much cheaper, safer, and there is no procedure needed to use one.

The consumer market for BCIs is currently relatively untapped, especially when one considers the potential for consumer applications namely with virtual reality (VR). There were approximately 50 million VR devices sold globally between 2014-2021 with 16.44 million of those sales in America alone [2] [3]. The market was valued at $21.83B in 2021, and is expected to grow by 15% from 2022-2030 [4]. VR promises a more immersive media/gaming experience, but the immersion is limited by the controls. EEG BCIs offer a unique solution to this problem, as the electrodes can be built directly into a headset and the ability to control the device with your thoughts is about as immersive as you can get.

The other major application of non-invasive BCIs is for people who have impairments in their motor function due to various conditions. Many of such people may not want or be able to undergo a surgical implantation of a BCI device, which limits their options for interacting with computers and the world around them. Non-invasive BCIs offer a more convenient and comfortable alternative that can enable them to communicate and control devices using only their brain activity.

B. Related Works

The notion of building EEG BCIs has been around for a while. Though consumer devices are currently uncommon, there are a handful of companies working on these devices. One such company, Interaxon Inc., recently launched a software development kit and EEG headband explicitly designed for use in VR systems [5]. Outside of VR and consumer devices, much work has been done in research settings for BCI game controller. Liao et al. [6] presented a BCI game controller using novel sensors in 2012. Advancements have also been made in improving the classification models powering BCIs. One of the most prevalent of such models is EEGNet [7], a convolutional neural network (CNN) architecture that manages competitive performance with significantly fewer parameters than comparable models. Another interesting model is EEG-Conformer [8], a convolutional transformer network that combines spatial-temporal convolutions, pooling, and self-attention to effectively classify EEG data.

C. Problem Definition

The combined human aid and consumer market potential for non-invasive BCIs is massive, yet these kinds of devices are still rare in the real world. We set out to build a robust simple game controller using a relatively cheap EEG, OpenBCI's Ultracortex Mark IV with the 8-channel Cyton board. Using this device, we aimed to build a binary controller and applied it to play a custom version of the game Flappy Bird.

Most effective research and production EEG BCI systems use much more expensive hardware, including more electrodes and electrodes of higher quality reducing the noise in the brain

signals. Due to budget constraints we were limited to cheaper hardware with lower fidelity signal. This issue leads to three guiding principles in building a BCI system: extensive signal processing to compensate for the noisy signal, using signals that are maximally reflected in the data as control inputs, and a careful training process to avoid overfitting noise patterns in the signal.

II. Methodology

A. Target Control Signal

At the onset of the project we explored multiple potential control signals. The first attempt was to use thought signals (i.e. thinking "jump"), however we quickly found there simply wasn't enough relevant signal in the EEG readings. Similarly, we experimented with motor imagery signals such as thinking about arm movements or jumping, but again these patterns were indiscernible in the data. Finally, we settled on physical motor movements. We considered using arm or leg movements, but in the joint interest of keeping the system maximally useful for those with disabilities and having as strong of a signal as possible, we decided to use blinking as the control mechanism.

B. Data Collection

For both data collection and control integration, we built a custom version of Flappy Bird written using PyGame. The EEG data stream was integrated into the game using Brainflow, and we built two game modes for interfacing with the device: data collection and BCI control. In both modes the headset is connected to and data is constantly streamed into a buffer from the device on launch. In data collection mode, the user plays the game using the spacebar to jump, and is instructed to blink anytime they press space. The program then listens for the spacebar press, and inserts a marker in the EEG data. On exit the data is formatted and written to a csv file. In BCI mode, the program evaluates each packet of 255 data points, feeding the data into the model to determine whether or not to jump.

Data is streamed at 250hz in packets of 255 readings, including voltage readings from each of the eight EEG electrodes, accelerometer data, timestamps, and some other unused readings. We only used EEG readings for our models. Importantly, readings from the EEG are scaled by a factor given by:

$$\frac{4.5V}{\text{gain}} \times \frac{1}{(2^{23}-1)},$$

where gain is a user-configurable value of: 1x, 2x, 4x, 6x, 8x, 12x, or 24x. For our work we used the maximum gain (24x), and therefore had to scale data by 0.02235 microVolts. The final term, 2^{23} is required because the Cyton board uses the ADS1299 chip, which is a 24-bit device, and outputs data in two's complement format [9].

C. Signal Processing

Signal processing was a crucial step in designing our BCI. EEG signals are measured as by placing electrodes on the scalp and recording the voltage differences between them (Figure 1). This raw voltage reading includes significant noise. If left unfiltered, the noise masks nearly all the patterns a model could use classify signals.

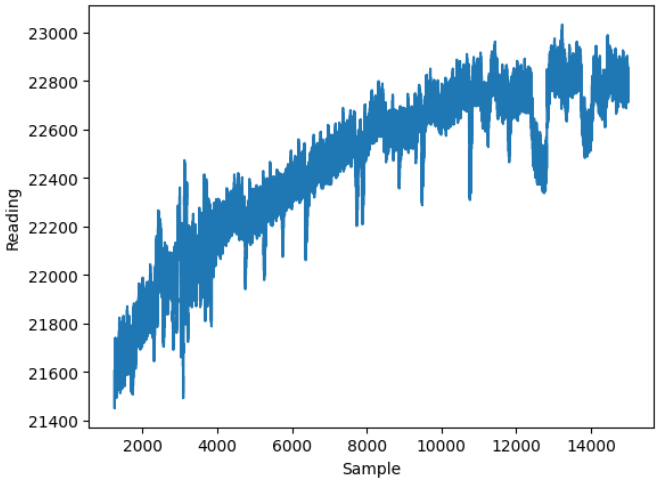

Fig. 1. Raw EEG data

Brain waves can be broken down by frequency (Figure 2), and motor signals dominantly reside in gamma (30hz+) waves. Therefore, signal processing techniques are applied to enhance the quality and extract meaningful features from the EEG signals.

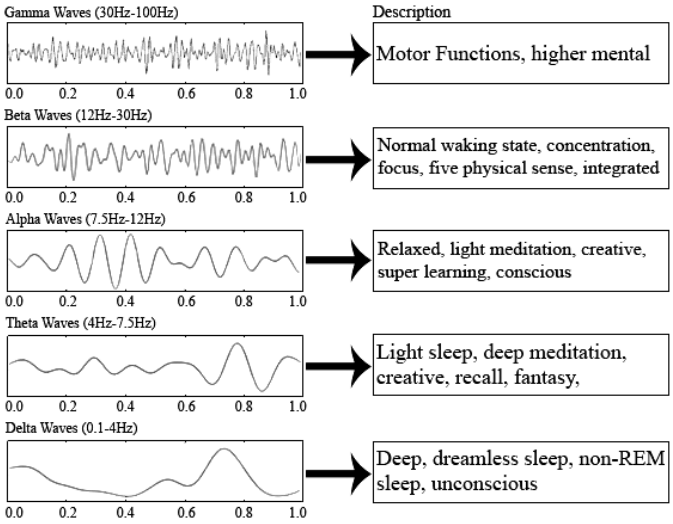

Fig. 2. Brain wave frequency ranges.

The first, and arguably most important, function applied is the Fourier transform, which decomposes signals into a sum

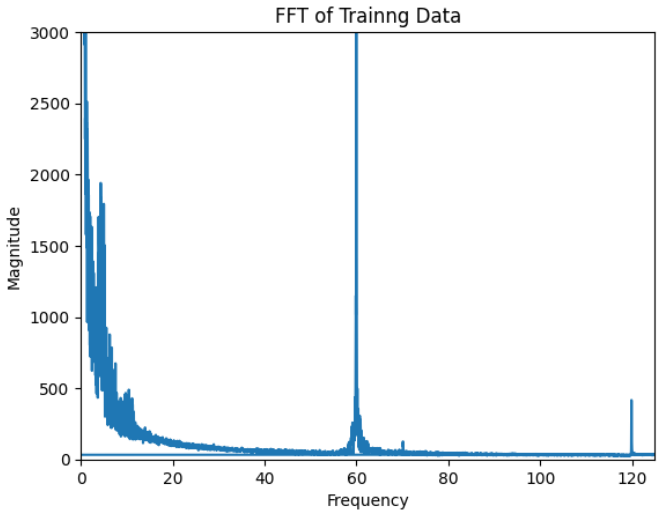

Fig. 3. Fourier transformed EEG data.

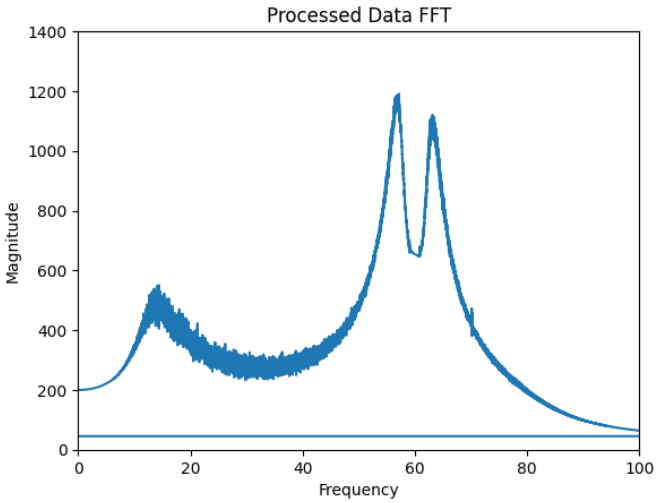

Fig. 4. Processed EEG data.

of overlapping sine waves of varying frequency. In practice we opted for `numpy`'s implementation of the fast Fourier transform (FFT) for its computational efficiency. By applying the FFT, one can identify the dominant frequencies or spectral components of the EEG signals, which reflect the different brain waves.

The result of applying the FFT to the raw signal is shown in Figure 3. In the plot there are three obvious spikes of noise, the most noticeable being the drastic spike at 60hz. This is caused by the background of AC power, which (in Canada) is 60hz. Also, as previously discussed, we are only concerned with signal in a certain frequency range. To address these issues, we apply a bandpass and a notch filter. The notch filter is a type of band stop filter that blocks a narrow band of frequencies, allowing all others to pass through. We applied this at 60hz with a notch size of 3, to knock out the AC power noise. The bandpass filter is used to filter for only the frequency range we care about: 13hz-80hz (i.e. filtering for only gamma and beta waves; while motor signals predominantly reside in gamma waves, empirically we found including beta waves gave better results). The filtered signal is shown in Figure 4.

A technique we experimented with but were unable to utilize effectively was band power, which quantifies how potent certain frequencies are within different ranges. Band power can be calculated by squaring the amplitude of each frequency component obtained by FFT. By measuring band power, one can compare the relative strength of different frequency bands across time or electrodes. Unfortunately, in this motor classification task band power didn't provide a useful feature and only degraded the performance of our models.

A final consideration for EEG signal processing is Nyquist frequency, which is the minimum sampling rate required to capture all the information in a signal without aliasing [10]. According to Nyquist theorem, one needs to sample at least twice as fast as the maximum frequency of interest. For EEG signals that range from 13 to 80hz, one needs to sample at least 120hz. However, since our EEG device has a higher sampling rate of 250hz this was not an issue for our BCI design.

D. Statistical Model

Due to the proximity of the eyes and related muscles to the brain, blinking produces a rapid voltage drop in the EEG data, shown in (Figure 5). This led us to trying a simple statistical classifier on the data. The classifier simply compares the rolling average signal strength of a 60 sample window to the average of the previous window. Because packets come in 255 sample bursts, we were able to run the classifier live with 4 windows to compare.

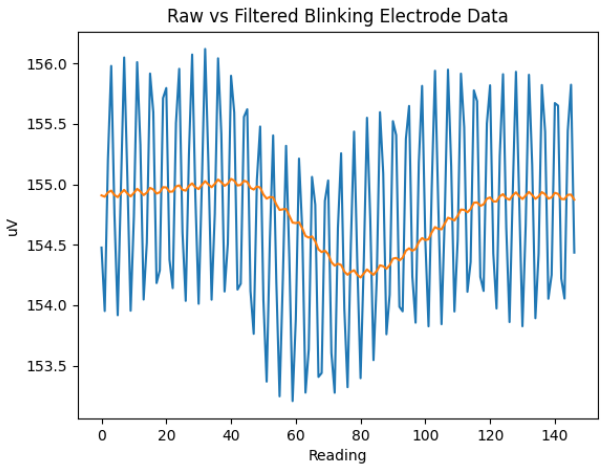

Fig. 5. Forehead electrode data recorded during a blink.

E. Convolutional Neural Network

The first ML model we built was a CNN based on the EEG-Net [7] architecture (Figure 6). We simplified the architecture, namely by reducing the convolutional layers from 2D to 1D in order to fit our collected data. This is composed of three convolutional layers, each with a kernel of 3 and immediately followed by batch normalization to stabilize training and add regularization helping the model to generalize. The second and third convolutional layers are followed with ELU activation, average pooling, and 25% dropout. Finally the outputs are passed through a fully connected layer with two outputs and softmax activation to give the prediction.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 3, 64)             192

batch_normalization (BatchN  (None, 3, 64)             256
ormalization)

conv1d_1 (Conv1D)            (None, 3, 64)             12288

batch_normalization_1 (Batc  (None, 3, 64)             256
hNormalization)

activation (Activation)      (None, 3, 64)             0

average_pooling1d (AverageP  (None, 1, 64)             0
ooling1D)

dropout (Dropout)            (None, 1, 64)             0

conv1d_2 (Conv1D)            (None, 1, 128)            24576

batch_normalization_2 (Batc  (None, 1, 128)            512
hNormalization)

activation_1 (Activation)    (None, 1, 128)            0

average_pooling1d_1 (Averag  (None, 1, 128)            0
ePooling1D)

dropout_1 (Dropout)          (None, 1, 128)            0

flatten (Flatten)            (None, 128)               0

dense (Dense)                (None, 2)                 258

activation_2 (Activation)    (None, 2)                 0

=================================================================
Total params: 38,338
Trainable params: 37,826
Non-trainable params: 512
_________________________________________________________________
```

Fig. 6. CNN model architecture, based on EEGNet.

F. Singular Vector Machine

The second ML model was a support vector machine (SVM) utilizing time series data. A rolling moving average was applied to filter out high-frequency oscillations. Using smoothed data, the SVM was trained to identify the distinct trough shape shown in Figure 5. Performing a binary classification on time series data allows us to identify blinks occurring in real-time with high accuracy.

III. Results

TABLE I
PERFORMANCE METRICS OF THE CLASSIFIERS.

Model	Accuracy	Precision	Recall	F1-Score
Statistical Classifier	81.36%	76.92%	57.97%	66.12%
CNN (overfit)	92.82%	93.42%	92.82%	92.80%
SVM	94.43%	92.85%	100%	96.30%

A. Statistical Model

The simple statistical classifier performed suprisingly well. It achieved an accuracy of 81.36%, and had very few false positives (only 5.45% of predictions). However, it noticeably lacked in its false negative rate at 13.18%. This high false negative rate is reflected in the low recall score of the model of 57.97%. In practice, this led to a relatively smooth gaming experience where the occasional input was missed but this was easily rectifiable by blinking again quickly. Along with good performance, this model had the added advantages of extremely quick prediction time and being entirely resilient to imbalanced data. This made it by far the quickest to build and had minimal impact on the performance (framerate) of the game.

B. Convolutional Neural Network

The CNN model had impressively good performance on validation data. It achieved 92.83% accuracy, and similarly high scores for all the other performance metrics. However, upon testing on data recorded on a different day, it was clear this model had severely overfit to its training data. On other datasets the model often had extremely high false negative rates (upwards of 90%) and was unusable other than briefly immediately after training.

C. Support Vector Machine

The SVM provided promising results across all metrics. Accuracy, recall, and F1 scores outperformed both the CNN and statistical classifier. The only metric in which SVM did not score highest was precision, in which the CNN outscored it by less than 1%. Furthermore, the model generalized well, identifying blinks with a cross-validation accuracy of 94.43%. Overall, the SVM provided strong results and generalization.

IV. Conclusion

We were successful in building an EEG BCI game controller. Of the models built, the SVM performed the best and both the SVM and statistical classifier worked as playable BCI control algorithms in practice. The CNN did not perform well or work in live testing due to a failure to generalize.

To effectively implement a CNN or other deep learning models, significantly larger datasets are required. Future steps for this project could include data collection across multiple people, spanning the course of several months. This process

could mitigate the overfit observed in our trials and would provide a more generalizable model.

Future steps for this project include increased control complexity. Using different actions to provide several control options to the user would provide a more ergonomic interface. Within the scope of this EEG's capabilities, it could be possible to provide two control signals through winking the right and left eyes, and possibly another movement with the arms or legs. This three-input system may cause a decrease in performance, but would provide the user with higher utility when interfacing with a computer.

Many options are possible to increase the performance of the BCI. First, using wet electrodes provides more accurate data. When using dry electrodes, interference from the user's hair and skin oils adds increased noise to the signal. Wet electrodes apply a conductive gel to the user's skin, which mitigates these sources of error. Secondly, using a higher quantity of electrodes would provide more detailed data. Tasks such as motor imagery and thought recognition are achievable using higher resolutions EEGs, ranging from 16 to 256 electrodes [11]. Additionally, electrode locations can be customized to provide signals from desired parts of the brain. For example, many motor signals originate from the frontal lobe of the brain. Re-locating electrodes to provide higher density over the frontal cortex could provide better results for motor movement classification. Exact electrode locations could be chosen to meet the needs of the user and the task at hand.

In short, the results achieved from this BCI project were promising. The goal of controlling a single-input video game was achieved with high consistency. Investigation into different control signals and new EEG technology could provide a more immersive interface for the user, and the development of this technology will enable those with disabilities to interact with the world in a more effective manner.

REFERENCES

[1] Neuralink, "Pager Plays MindPong," Neuralink, Apr. 18, 2021. https://neuralink.com/blog/pager-plays-mindpong.

[2] M. M., "29 Virtual Reality Statistics to Know in 2023," Leftronic, Mar. 7, 2023. https://leftronic.com/blog/virtual-reality-statistics/.

[3] Statista, "AR/VR headset shipments worldwide 2019-2023," Statista, Jan. 25, 2022. https://www.statista.com/statistics/653390/worldwide-virtual-and-augmented-reality-headset-shipments/.

[4] Grand View Research. "Virtual Reality Market Size, Share & Trends Analysis Report By Technology (Semi & Fully Immersive, Non-immersive), By Device (HMD, GTD, PDW), By Component (Hardware, Software), By Application, By Region, And Segment Forecasts, 2022 - 2030." Grand View Research. https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market.

[5] "Interaxon Inc. (MUSE) launches new VR SDK & VR compatible EEG band to support innovations in brain health through Biosensor integration into VR & AR applications in the metaverse," Business Wire, Mar. 3, 2022. https://www.businesswire.com/news/home/20220303005358/en/Interaxon-Inc.-Muse%C2%AE-Launches-New-VR-SDK-VR-Compatible-EEG-Band-to-Support-Innovations-in-Brain-Health-Through-Biosensor-Integration-into-VR-AR-Applications-In-The-Metaverse.

[6] Liao, LD., Chen, CY., Wang, IJ. et al. Gaming control using a wearable and wireless EEG-based brain-computer interface device with novel dry foam-based sensors. J NeuroEngineering Rehabil 9, 5 (2012). https://doi.org/10.1186/1743-0003-9-5.

[7] V. J. Lawhern et al., "EEGNet: A Compact Convolutional Neural Network for EEG-based Brain-Computer Interfaces," arXiv preprint arXiv:1611.08024v4 [cs.LG], Nov. 23, 2016.

[8] Y. Song, Q. Zheng, B. Liu and X. Gao, "EEG Conformer: Convolutional Transformer for EEG Decoding and Visualization," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 31, pp. 710-719, 2023, doi: 10.1109/TNSRE.2022.3230250.

[9] OpenBCI, "Cyton Data Format," OpenBCI, 2023. https://docs.openbci.com/Cyton/CytonDataFormat/.

[10] P. Colarusso, L. H. Kidder, I. W. Levin, and E. N. Lewis, "Raman and Infrared Microspectroscopy," in Encyclopedia of Spectroscopy and Spectrometry (J. C. Lindon, G. E. Tranter, and J. L. Holmes, Eds.), 2nd ed., vol. 3, Oxford: Elsevier, 2010, pp. 1945-1954.

[11] L. Xiangmin, L. Jianwei, Z. Yifei, H. Ziqian, H. Yimin, "A Motor Imagery Signals Classification Method via the Difference of EEG Signals Between Left and Right Hemispheric Electrodes" in Frontiers in Neuroscience, vol. 16, 2022

Ethical Evaluations of Institutional Actors Deploying AI Tools

Jack Porter
Queen's University
18jp11@queensu.ca

Sara Laker
Queen's University
21sjl8@queensu.ca

Alex Nguyen
Queen's University
22thn@queensu.ca

Avery Goreglad
Queen's University
avgoreglad@gmail.com

I. Introduction

AI ethics is increasingly recognized as a vital component of understanding how best to utilize AI. As AI grows in applications, there is a greater need to understand the ethics of each particular deployment and whether or not it aligns with our societal conception of ethics. In this paper, we argue that to fully understand the ethics of AI, it is not sufficient to only look at the technology. Rather, the current social, political and economic institutions using the AI must be evaluated along with the technology, as evidence suggests the (un)ethical deployment of AI tools stem not from the technology itself but from the institutional actor utilizing the AI algorithms.

What follows is a brief explication of this theory which suggests understanding the ethics of technology means expanding beyond the technology and looking at the institutions which are increasingly harnessing the power of AI. Then, we present three case studies which show how unethical AI is not due to the technology but the institutional actor (hiring practices, the policing system, or the welfare state) using the technology. The paper concludes with some broader reflections on how we can evaluate these institutions so that they utilize AI in a more ethical manner.

II. Theoretical Framework

As Artificial Intelligence grows in its possible applications, its ethical implications expand simultaneously. The increasing popularity of ethical AI as a critical component of utilizing technology for general societal advancement is promising. AI no longer exists in isolation, harboured in academia or within dominant technology companies, but now seeps into various aspects of social life. This growth in applications is exponential as human ingenuity continues to find novel ways of applying AI algorithms for diverse, disruptive purposes. Furthermore, with this en masse application of AI algorithms into social institutions, this paper proposes we look to ethically evaluate the various institutions looking to deploy AI tools, as this has potential consequences for the ethical status of the AI technology itself.

In the seminal work on AI Ethics titled Do Artifacts Have Politics? Philosopher Langdon Winner looks to understand how political relations and social values may be embedded into the technologies we use in society. In other words, the technology we develop and deploy is socially contingent, like our understanding of taxation and welfare. Just as these social functions and services reflect the values we hold in society, so too does our technology. In the article, Winner writes, "those who have not recognized the ways in which technologies are shaped by social and economic forces have not gotten very far" [1]. What this means is that if we are to take AI ethics seriously, and understand the complete ethical picture of technology in society, then we need to factor in the existing social and economic values in our society within this picture. In this case, it is not about looking at the social impact technology may have and evaluating the ethics which emerge. Instead, it is about understanding how social values inform the development of current technology and how AIs' ethical status relates to the dominant social and economic forces within society. Whenever these forces are ethically suspicious, the technology has a strong chance of being no better. The ethical status of AI is, in some cases, downstream from the ethics of society in general. The danger we must avoid is AI tools aiding and abetting these unethical institutional practices, thus making the AI itself unethical. What follows are three case studies that highlight how the unethical practices of an institution directly lead to the unethical use of AI. With this, we should not direct our attention to the technology but to the institutions that utilize it. And the solutions don't necessarily lie in the technology but in specific institutional reforms to address these unethical practices.

III. Case Studies

A. Discriminatory AI Hiring Tools

Many people have heard the saying that robots and artificial intelligence are "going to take your job," but instead of taking, it might just be what is preventing you from even getting that position. Organizations increasingly turn to artificial intelligence programming for help when looking for the person best fit for the job.

Recently, AI has been used in many capacities for various things, such as identification, analysis, and assessment. There is a promise that comes with using these programs; that it can find the most fitting and capable person that matches the job description with both speed and efficacy [3].

These tools can access human databases for capital decisions and give feedback to applications on their strengths/weaknesses, development pointers, and career and organizational fit advice [4]. Through this same process, AI can search through candidates' social media postings, any

and all accounts created under their names/information, do a linguistic analysis of both speech and writing samples, conduct video analysis for tone, facial expressions/emotions, and body language/nonverbal cues and behaviours, to name a few [2]. Having tools like this in the hands of organizations can be seemingly revolutionary; however, such unprecedented power can have underlying effects that don't meet the eye.

While this technological advance is undoubtedly exciting to see where it can impact organizational practices, many unanswered questions and concerns are related to ethics, privacy, and legality [4]. With barely any oversight from the government or ethics boards, AI companies can create and use developed software that can make decisions on just about anything without having to run it by anyone to ensure that the programs are not encoded with conscious or unconscious internal and structural biases [3]. Many other scientifically derived assessment tools have been approved due to their reliable relationships between potential candidates' "scores" and their job performance [2]. Tests for both cognitive ability and intelligence have proven time and time again to be reliable and correct indicators of job success for a wide range of occupations. The problem arises initially with concerns around discriminatory practices and the possible impacts that can arise for equal opportunity for applicants. These AI programs can now take learned processes fed to them by programmers to answer questions that cannot be asked in interviews (ex. Being able to determine if someone is mentally ill due to a visual cue, verbal inference/the way they speak or the dialect that they use on where they are from, or social media post) [2].

With these preliminary hiring programs being newer and having less information than earlier programs, there are many ethical concerns to confront. Instead of being well-tested and scientifically proven work methods, they are technological innovations curated for employers with the promise of ease and reliability. Due to Acts like the Americans with Disabilities Act (ADA) and the Accessible Canada Act, employers cannot inquire about physical and mental disabilities when doing initial candidate assessments [2]. For example, someone with a speech impediment or a stutter would be ruled out through linguistic analysis in a program. Same goes for someone who may have a thick accent, not be a nature speaker of the language, or someone who talks a certain way because of the neighbourhood they are from - all traits that could impair them from moving forward in the interview process.

The growing appeal of AI products that promise efficiency and reliability for any process one needs is undeniable [4]. With such significant steps being taken to improve workplaces and become an integral part of the hiring structure, companies must become aware of the possible ethical and moral concerns that can come from this.

B. Racial Bias in AI Policing Tools

In recent years, the utilization of AI has rapidly expanded, and its applications are vast. However, the utilization of AI in policing has brought up biases and ethical dilemmas surrounding its decisions and outcomes. Police implement AI through two types of technology: facial recognition and predictive policing. Both have different tradeoffs, biases, and ethical implications.

Facial recognition technology is used for security through the form of identification surveillance. An analysis of its effectiveness and accuracy within policing is well documented in two studies: "Facial recognition in policing: A study of the Metropolitan Police Service" [9] and "The Perpetual Line-Up: Unregulated Police Face Recognition in America" [10]. The first analyzed data from the London Metropolitan Police and found that the use of facial recognition technology led to a 20% increase in the number of suspects arrested for crimes. This suggests that the technology has the potential to enhance the effectiveness of policing by helping to identify suspects. The study also found that the technology was particularly effective in identifying suspects of serious crimes, such as murder and attempted murder. This suggests that facial recognition technology may be particularly useful in investigations involving high-stakes criminal activity. However, the study also highlighted some of the limitations of using AI in policing. One potential limitation of the technology is that it may be less effective in identifying suspects who have not been previously arrested or have not had their photographs taken by the police. Additionally, facial recognition technology may be less effective in identifying suspects from certain demographic groups, such as individuals with darker skin tones or older individuals, which raises concerns about bias in the technology. The second study reported that many facial recognition systems used by law enforcement agencies in the United States have significant racial and gender biases. This means that individuals from certain demographic groups, such as people of colour and women, are more likely to be misidentified by technology than individuals from other groups. The cause is theorized through an analysis of the 2006 NIST competition. Which discovered that facial recognition algorithms developed in East Asia performed better on East Asians while algorithms developed in Western Europe and the U.S. performed better on Caucasians. This raises serious concerns about the potential for wrongful arrests of minority ethnic groups within a population.

Predictive policing uses algorithms to identify individuals who are likely to commit crimes in the future and targets them with increased surveillance. Analysis and criticism of Predictive policing are found in two studies: " Evaluating the effectiveness of predictive policing: A randomized controlled trial" [11] and "Policing in the Era of Big Data" [12]. The first study identifies predictive policing as a crime-fighting strategy that uses data and analytics to identify potential crime hotspots and allocate resources accordingly. The study used a randomized controlled design, with half of the city's police beats receiving predictive policing and the other half serving as the control group. The study found that the use of predictive policing led to a significant reduction in the number of total crimes and Part 1 crimes (which include serious crimes such as homicide, rape, robbery, and aggravated assault) when compared to the control group. Specifically, the study found

that in the areas that received predictive policing, there was a 16% reduction in total crimes and a 25% reduction in Part 1 crimes when compared to the control group.. One of the key findings of the study is that the use of predictive policing did not lead to a significant increase in arrests or citations. This suggests that the reduction in crime was not due to an increase in arrests or citations but rather an increase in proactive policing, such as increased patrols in high-risk areas. This is an important finding, as it suggests that predictive policing can be an effective crime-fighting tool without relying on increased arrests or citations. This study provides evidence that predictive policing can be an effective tool in reducing crime, and it is important for law enforcement agencies and researchers to continue to critically examine the efficacy of predictive policing.

The second study highlights the limitations of using AI in policing. The author notes that while predictive policing has been praised for its ability to reduce crime rates, the effectiveness of the technology is still largely unproven. One of the key limitations of predictive policing highlighted in the study is the potential for bias in the algorithms used. The study notes that the data used to train predictive policing algorithms is biased because embedded in the data possesses information that is affected by past policies that are discriminatory and unjust in nature, which leads to unfair targeting of certain communities. Additionally, the study highlights the lack of transparency in the algorithms used for predictive policing and the lack of oversight, which may lead to abuse of the technology. The study concluded that there is a need for more public engagement and dialogue around the use of predictive policing, as it has the potential to impact policing and criminal justice significantly. As well as the importance for policymakers, law enforcement agencies, and researchers to continue and critically examine the use of AI in policing and consider the potential benefits and limitations.

In conclusion, unconscious biases in the use of AI in law enforcement can bring about significant ethical problems and result in unjust and unfair consequences. The ethical shortcomings of these organizations stem from various elements, including the algorithms' fairness and accuracy, the lack of transparency and responsibility in the creation process, and the possibility of promoting discrimination and maintaining existing biases. To tackle these concerns, it is crucial for these organizations to be open and accountable in the creation and use of AI algorithms and to put in place safeguards to stop and rectify any biases that may exist. This will guarantee the responsible and ethical usage of the technology and its fair benefits for all, regardless of demographic factors such as race, gender, age, or others.

C. AI and Welfare State Automation

AI has been used in the welfare system to automate, identify, and enhance functionality. But what makes AI so versatile may also lead to some unethical outcomes such as personal bias reflected in the algorithm, problematic selection criteria, and breaches of privacy. AI shows promise in relieving the welfare state of some administrative burdens, but this comes with noteworthy ethical tradeoffs.

AI is a promising tool for the welfare state as it is cost-effective and can derive purposeful insights. For example, the municipality of Gladsaxe in Copenhagen, Denmark, is designing a system to flag households potentially responsible for child abuse. They do this by condensing and sifting through information on residents' health records and employment information. [5] Large amounts of data are rapidly accessed by a system over which welfare administrators have control. In Gladsaxe's child abuse intervention system, lists of information on Danish residents can be accessed and shared between municipalities and may potentially make its way to the national government. [5] The increased reliance on the algorithms has led to less human engagement with these cases. An issue with this is the potential for breaches of privacy among governments using these large data sets. Privacy issues dominate discussions of AI ethics and there is a need to have this discussion specifically for data collected by the welfare state.

In the Gladsaxe example, AI is used to identify households that were potentially abusive towards children, but it was later discussed the same algorithm could be used to identify people based on alternative criteria. An ethical issue arises when the same system used to detect victims of child abuse is also exploited for other, more nefarious purposes. Users may exploit AI under the disguise of righteousness and virtue, and it bolsters another concern about discrimination based on specific criteria. [6]

By training the AI to identify samples in a population using a well-labelled set of data, it begs the question of how an administrator chooses these recognition patterns for the algorithm. What is contained within this pre-labelled data? What are the consequences of repeating these patterns?

The UK has currently spent millions creating welfare robots to enhance welfare service delivery, replacing human employees to reduce error and latency. Likewise, the government of India issued unique 12-digit identification numbers to country residents in the world's largest biometrics experiment to grant individuals welfare services based on health records, financial standings, and other factors. [7] Here, welfare administrators are attempting to replicate human actions, ideas, and behaviours. However, a drawback is that some systems have focused on this aspect so much that they rely on it to replicate human judgement. Some administrators are beginning to sacrifice their agency to AI rather than enhancing it, granting machine-learning systems the ability to judge whether an individual deserves welfare service or not. This is fatal when individual biases are reflected in welfare algorithms. These biases are prevalent in how AI functions: in the UK, welfare robots may replace low-income employees in the welfare sector. Further, low-income individuals in India are at risk of losing their identities in the event of security breaches or technological errors. This would be catastrophic for residents who rely on their 12-digit ID to access welfare services [8].

While AI frameworks have been enhancing the ability to

effectively meet societal demands, some welfare administrations have silently begun to use them to remove human responsibility and care. AI is a powerful tool that enhances the human welfare system through automation, identification, and strengthening of human agency. Nevertheless, there are societal consequences when AI is used with specific biases and values, such as breaches of privacy, welfare eligibility, and the removal of human responsibility as a result. Welfare concerns the state of everyone within a specified community: AI is only as ethical as the users who wield it.

IV. Reflections on Ethical AI

In the conclusion of his article, Winner writes, "to understand which technologies and which contexts are important to us, and why, is an enterprise that must involve both the study of specific technical systems…as well as a thorough grasp on the concepts and controversies of political theory" [1]. This means we must look at technology as political, not isolated from political forces. This is not to say all technology is doomed to be politically charged and thus never oriented towards universally agreeable purposes, but to say that tech exists within our social world and thus must be evaluated as such. Increasing awareness of the political elements of AI will greatly assist in understanding its ethics. In the cases we have shown, the ethics of AI are a reflection of existing ethical issues in our society; our technology is intimately related to these political issues. If we deem AI to be unethical, it doesn't follow that discontinuing the AI will make society more ethical, as the ethical status of the AI is contingent on the ethics of society. Accepting this is a strong first step in making technology more ethical.

References

[1] W, Langdon. "Do Artifacts Have Politics?" Daedalus 109, no. 1 (1980): 121–36. [Online]. Available http://www.jstor.org/stable/20024652.

[2] B. Dattner, T. Chamorro-Premuzic, R. Buchband, and L. Schettler, "The legal and ethical implications of using AI in hiring," Harvard Business Review, 25-Apr-2019. [Online]. Available: https://hbr.org/2019/04/the-legal-and-ethical-implications-of-using-ai-in-hiring. [Accessed: 11-Feb-2023].

[3] C. Pazzanese, "Ethical concerns mount as AI takes bigger decision-making role," Harvard Gazette, 26-Oct-2020. [Online]. Available: https://news.harvard.edu/gazette/story/2020/10/ethical-concerns-mount-as-ai-takes-bigger-decision-making-role/. [Accessed: 11-Jan-2023].

[4] N. Parikh, "Council post: Understanding bias in AI-enabled hiring," Forbes, 14-Oct-2021. [Online]. Available: https://www.forbes.com/sites/forbeshumanresourcescouncil/2021/10/14/understanding-bias-in-ai-enabled-hiring/?sh=3ae4d7867b96. [Accessed: 26-Feb-2023].

[5] J. Mchangama, "The Welfare State is Committing suicide by Artifical Intelligence", Foreign Policy. December 25 2018. [Online]. Available: https://foreignpolicy.com/2018/12/25/the-welfare-state-is-committing-suicide-by-artificial-intelligence/ (Accessed: 18-02-2023)

[6] P. Alston, "World stumbling zombie-like into a digital welfare dystopia, warns UN human rights expert", United Nations. October 17 2019. [Online]. Available: https://www.ohchr.org/en/press-releases/2019/10/world-stumbling-zombie-digital-welfare-dystopia-warns-un-human-rights-expert (Accessed: 18-02-2023)

[7] E. Pilkington, "Digital Dystopia: How Algorithms Punish the Poor", The Guardian. October 14 2019. [Online]. Available: https://www.theguardian.com/technology/2019/oct/14/automating-poverty-algorithms-punish-poor (Accessed: 18-02-2023)

[8] R. Yang, "Who Killed Manjhi? — A Look at the Digital Social Benefits Programmes", Medium. November 13 2019. [Online]. Available: https://medium.com/@rachelyeung_32216/who-killed-manjhi-a-look-at-the-digital-social-benefits-programmes-283feecac07d (Accessed: 18-02-2023)

[9] Metropolitan Police "Facial recognition in policing: A study of the Metropolitan Police Service," ND [Online] Available: https://www.met.police.uk/advice/advice-and-information/fr/facial-recognition

[10] C. Garvie, A. Bedoya, and J. Frankle, "The Perpetual Line-Up: Unregulated Police Face Recognition in America" Perpetual Line-Up October 16, 2016. [Online] Available: https://www.perpetuallineup.org/

[11] M. Maximino, "Evaluating the effectiveness of predictive policing: A randomized controlled trial," The Journalist's Resource, November 6, 2014. [Online] Available: https://journalistsresource.org/criminal-justice/predictive-policing-randomized-controlled-trial/

[12] G. Ridgeway, "Policing in the Era of Big Data," Annual Review of Criminology, September 27, 2017. [Online] Available: https://www.annualreviews.org/doi/pdf/10.1146/annurev-criminol-062217-114209

Forecasting Stock Price Movement with Google Trends Data

Thomas Tesselaar
Queen's University
thomas.tesselaar@queensu.ca

Gary Farberov
Queen's University
19gaf3@queensu.ca

Maddy Chapnik
Queen's University
20mnc4@queensu.ca

Anna Lou
Queen's University
21al98@queensu.ca

***Abstract*—Accurately predicting how a stocks' price will behave is one of the most examined problems in finance. More accurate prediction methods have the potential for large financial gain. At the same time, Google Trends can serve as a window into people's interests and a reflection of broad trends. We hypothesized that some search terms may be useful as predictors. To do this, we looked at three stocks and found search terms that correlated with each one. We built a predictor to predict price movement, which outperformed the market for all three equities. Finally, we simulated trading with our model dictating trading decisions and found we returned higher than the market for all three equities we tested.**

I. Introduction

A. Motivation

The question of how to accurately predict how a stocks' price will behave is one of the most examined problems in finance. In the U.S. alone, there are over 7,000,000 employees in the finance and insurance industries [1], of whom at least 100,000 [2] are involved in dealing of debt and securities. More accurate prediction methods have the potential for large financial gain. At the same time, Google Trends can serve as a window into people's interests and a reflection of broad trends. It is an open question as to whether it can be useful in predicting the behaviour of a stocks price.

We aim investigate the possibility of a relationship and to build and evaluate a predictor for stock price movement using Google Trends data.

B. Related Works

This project was intended to build on a paper by Dartanyon Shivers [4] which explored the relationship between Google Trends data and stock price data. That paper was a surface level dive into the relationship and found some correlation but didn't examine a significant number of terms, or if search data could be be used as a predictor. The paper concluded that there exist searches whose popularity correlates with stock prices over a given period but left it as an open question whether or not we could use the popularity of searches to predict future stock price movement.

C. Problem Definition

We hypothesize that some search terms may be useful as predictors, for example, an increase in searches for electric vehicles (EVs) may preclude increased demand and and increase to Tesla's share price.

We want to test this hypothesis on three different equities, the S&P/TSX Composite index (ticker ^GSPTSE), an exchange traded fund (ETF) which tracks a aggregate of 60 major stocks traded on the Toronto Stock Exchange (TSX), Apple, Inc. (ticker AAPL), and Tesla, Inc. (ticker TSLA). We also want to test if this would be effective as a trading strategy.

For the predictor to be considered successful, it must outperform the market defined by the predictor being able to predict a rise/fall in an equities price better than predicting a rise every week. We define an effective trading strategy as one which yields higher than market returns. For our modelling to be considers an effective trading strategy, it should provide a higher return by buying when the predictor says the stock will rise and selling when it predicts the stock will fall, and comparing this to the return had we bought the stock at the beginning of the time period and held it for the duration of the model.

II. Methodology

A. Data

There were two primary sources of data for this project, corresponding to the two types of data that were required. Yahoo! Finance was used to source all our stock data. Yahoo! Finance allows users to download any company or ETF's stock price history over any length of time beyond 1962. This includes adjusted price, taking into account any stock spilts.

Google Trends was used to source data on search terms. Google Trends randomly samples from its log of searches to determine a particular search term's relative frequency over time. Google automatically normalizes this on a scale of 0 to 100 (so if the search term "tomatoes" peaked at 0.005% of all searches then that point in time would have a value of 100 and a point in time where "tomatoes" was 0.0025% of all searches would have value 50). Google limits frequency of historical search term data to weekly when looking at more than 9 months of data. Google also provides the ability to specify regions for the searches. Google has no API's to automate the collection and usage of search term data. There is an unoffical API, PyTrends however this was outdated and we encountered issues so all data had to be manually downloaded which was a lengthy and timeconsuming process.

Both search data and stock data are noisy however looking at weekly time blocks provided a natural smoothing. We

looked at 10 years (521 weeks) of data for all of our data. For some terms we looked at Global, US, and Canadian data and found that the difference was marginal. It is important to note that there is also some small variation in the data because of the way it is randomly sampled however in our trials we found this difference was never more than 3 and was on average 0.72.

B. Measuring Correlation

We used two different methods to measure the correlation between some given search term and a stock. The first method was to find the Pearson correlation coefficient, which for a search term X and stock Y is denoted

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}, \tag{1}$$

where $\text{cov}(X,Y)$ is the covariance of X and Y and σ_X represents the standard deviation of X. This correlation is bounded above by 1 and below by -1. $|\rho_{X,Y}| = 1$ implies perfect correlation (the relationship between X and Y can be described linearly with all points exactly on the line) with the sign equal to the sign of the linear relationship, and $\rho_{X,Y} = 0$ implies no correlation (no linear relationship between X and Y). This is a standard correlation measure however with our data it has a weakness in that stock prices tend to increase over time but search terms are bounded and do not (in general) have a tendency to increase.

The second measure was to directly compare whether in a given week did both the stock and search term move in the same direction (did the frequency of searches and the stocks price both go up/down). We denote this method $\phi_{X,Y}$ for some search term X and stock Y. For a given week, if they both moved in the same direction this was given a score of 1, if they moved in opposite directions this was given a score of 0. This was then divided by the number of samples to get a value between 0 and 1. A value of 1 represents perfect correlation (movement direction is always identical), a value of 0 represents inverse correlation, and a value 0.5 represents no correlation (it is completely random whether they move in the same direction). This method was similar to the method proposed by Shivers [3].

Using their method, they confirmed that stocks then to behave randomly relative to each other. In their work, 100 stocks were compared against each other resulting in a mean correlation of 0.5 and a standard deviation of 0.03.

C. Data Pre-processing

From the stock data we derived our target attribute 'price movement', which takes on value 1 if the stock went up and -1 if the stock went down. It would take on value 0 if the stock didn't move however the were no instances of this in our data. We also derived features for each search terms that we used for predicting, and normalized our input data. We spilt our data sets into a randomly sample test set with 25% of our data and the remaining 75% as the training set.

D. Modelling

A multi-layer perceptron (MLP) is a type of artificial neural network. It is composed of three or more layers; a single input layer, one or more hidden layers, and a single output layer. Figure 1 shows an example of an MLP with two hidden layers. The input layes represents the data we know, that we are giving to the model. Each hidden layer contains some number of neurons, represented by the circles in the figure. Each neuron in the weighted layer receives input signals from the neurons in the previous layer, performs a weighted sum of these inputs, and then applies an activation function to produce its output. The activation function introduces non-linearity into the model, allowing the MLP to learn complex patterns and relationships in the data. The output layer also contains neurons in the same fashion as the hidden layers. The number of neurons in the output class corresponds with the number of classes you have that you are trying to predict. For some given input data, the models prediction is whichever neuron in the output layer has the highest value.

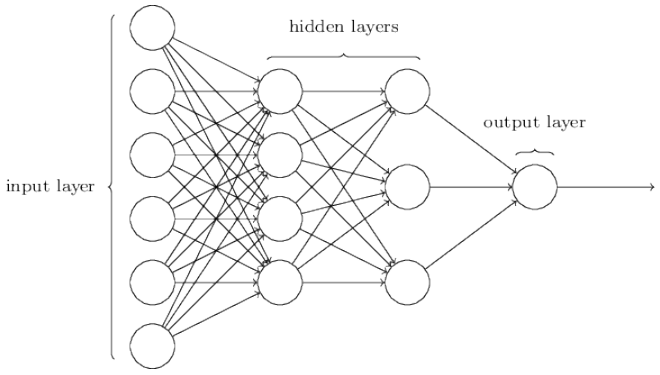

Fig. 1: Example of an MLP with two hidden layers.

The way the model learns is through by tweaking the weights and bias of each neuron. When training the model, every time the model makes a prediction, there is a loss function, which I will denote J, is used to "measure" how far the prediction was from the expected result. We want to minimize J, i.e. get our guesses closer to the expected result. To do this, a process called backpropagation (2) is used which takes the partial derivative of each weight. Using gradient descent tweaks it slightly in that direction. The equation for backpropagation is shown below where w_{ij} is the weight we are tweaking (weight i of neuron j), o_j is the output of the neuron (after the activation function), and z_j is the weights sum of inputs to the neuron ($z_j = \sum w_{kj} x_k$).

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial o_j} \cdot \frac{\partial o_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}}, \tag{2}$$

We chose to use an MLP as our predictor. Our input layer consists of search data and associated features. Our output layer contains two classes, corresponding with the price rising and the price falling. The size of our data was small enough that a neural network could be trained with minimal computing power and an MLP can handle non-linear

relationships through its multiple hidden layers, which allow them to capture complex patterns and interactions within the data.

III. Results

For each stock we chose 18-20 search terms that might be relevant and measured their correlation. For each term we found $\rho_{X,Y}$ and $\phi_{X,Y}$ then shifted $\phi_{X,Y}$ and stretched it to have the same range (and similar implications for the same values) as $\rho_{X,Y}$. We then took the Euclidean norm ($||(\rho_{X,Y}, \phi_{X,Y})|| = \sqrt{\rho_{X,Y}^2 + \phi_{X,Y}^2}$) and filtered out terms whose correlation was below 0.6. This left us with 5-8 terms for each stock.

Using the data from the search terms that had weighted correlation ≥ 0.6 with their respective stock, we cleaned and fed this data into our model. Table I below shows the results, where the 'market' column refers to the percentage of the time that the stock goes up, or what your accuracy would be if you predicted the stock price would rise every time.

TABLE I: Table showing results from our modelling

Stock	Market	Accuracy	Precision	Recall
^GSPTSE	57.36%	57.81%	55.26%	67.74%
AAPL	57.36%	64.06%	63.41%	76.47%
TSLA	56.45%	58.46%	61.29%	76.00%

All three models managed to predicted better than the market. All three also displayed a bias towards a higher recall, i.e. a preference for false positives or predicting that the stock would rise. The model for AAPL outperformed by a larger model than the other two. We hypothesize that this could be due to the fact that Apple's performance as a company is largely determined by consumer demand which could be reflected in Google searches.

This modelling alone does not answer the question of would the model outperform the market as an investment strategy. To simulate this, we tried buying when the model predicting the stock price would rise and selling when it predicted it would fall. The results of this analysis are shown in Table II below.

TABLE II: Table showing actual return alongside modeled return

Stock	Actual 10Y Return	Simulated 10Y Return
^GSPTSE	58.44%	79.74%
AAPL	995.7%	1418.7%
TSLA	7973.4%	10448.6%

All three stocks outperformed the market when trading according to our model, with AAPL again showing the biggest relative out-performance. The graphs show a comparison of our model against a buy and hold strategy, with our model in orange labelled "strategy", and buy and hold in blue. The three graphs, in order, are GSPTSE, AAPL, and TSLA.

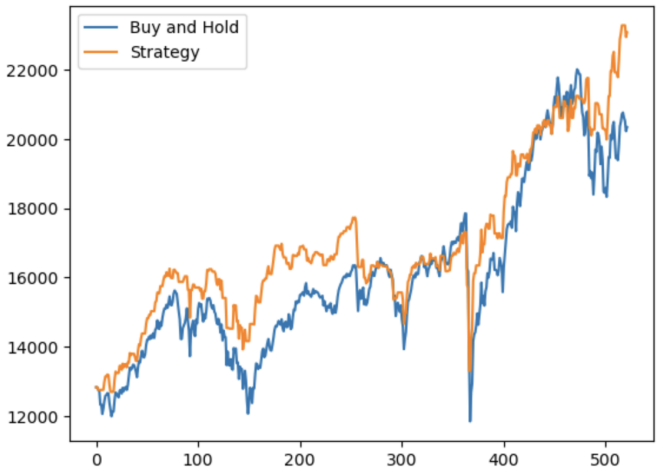

Fig. 2: Our model vs. buy and hold for GSPTSE.

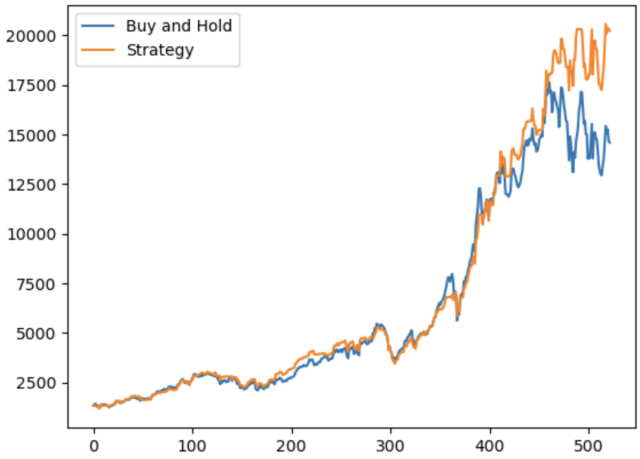

Fig. 3: Example of an MLP with two hidden layers.

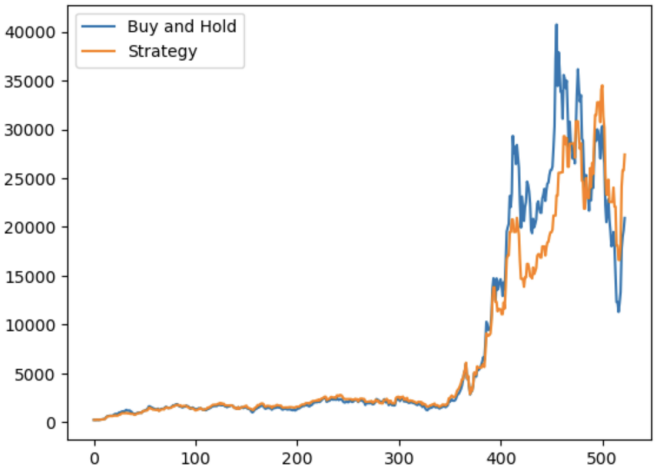

Fig. 4: Example of an MLP with two hidden layers.

IV. Conclusion

We tested our hypothesis and were able to successfully build predictors for all three equities we tested. Those models also

"traded" better than the market.

Some avenues for further investigation would be changing the data frequency from weekly to monthly which could smooth out some noise, or alternatively changing the data frequency to be daily. As well, different models could be tried than the single one we built. We would also recommend that anyone trying to duplicate our results look for a way to automate the Google Trends data collection process. In our calculations for investment return we did not consider dividends which could be included in future analysis.

It is important to note that historical correlation does not imply future correlation. Additionally, we are not qualified to recommend stocks or investment advice, and are merely highlighting the potential use of Google Trends in market analysis. Code used for this project can be found in our GitHub. This may not be exhaustive of all code since some analysis was done on local devices.

REFERENCES

[1] "Finance and Insurance in the US - Employment Statistics, "https://www.ibisworld.com/industry-statistics/employment/finance-insurance-united-states/

[2] "Investment Banking & Securities Dealing in the US, "https://www.ibisworld.com/industry-statistics/employment/investment-banking-securities-dealing-united-states/

[3] "Methodology and Sources, " https://help.ibisworld.com/s/article/methodology-and-sources

[4] D. Shivers, "Exploring the relationship between Google Trends data and stock price data, " https://www.ccom.ucsd.edu/~cdeotte/papers/GoogleTrends.pdf, unpublished.

GenomeAtlas

Ammar Lakdawala
Queen's University
19al61@queensu.ca

Emily Jiang
Queen's University
19eaj2@queensu.ca

David Hoernke
Queen's University
20dash@queensu.ca

Ian Fairfield
Queen's University
20idf@queensu.ca

Steven Zhang
Queen's University
19sz99@queensu.ca

***Abstract*—In this article, we explore the feasibility of leveraging natural language processing (NLP) to develop a search engine that can summarize and answer questions about relevant plant genomes from articles in Pubmed. To achieve this, we developed a high-quality plant-phenotype relationship (PPR) corpus containing information derived from 20,000 PubMed abstracts. Our goal is to demonstrate the feasibility of using GPT to summarize articles and provide a valuable resource for researchers in the bioinformatics industry. We also discuss Textpresso Central, a similar tool, and highlight the challenges faced by researchers in the field of plant genome research. Our aim is to assist researchers in efficiently extracting meaningful information from unstructured textual data by using NLP to identify articles that are most relevant to the user's search query.**

I. Introduction

A. Motivation

In recent years, the bioinformatics industry has shown a growing interest in leveraging natural language processing (NLP) to extract valuable information from unstructured textual data. To aid researchers in this field, we are exploring the feasibility of developing a tool that uses GPT to summarize articles from Pubmed about plant genomes. Our goal is to create a search engine that can both summarize and answer questions about relevant plant genomes, thereby enabling researchers to be more productive.

Existing literature contains valuable information about relationships between plants and phenotypes that researchers can use to develop NLP models. However, there is currently no appropriate corpus available to train and evaluate these models specifically for plants and phenotypes, which poses a challenge for researchers. Our project aims to provide a solution to this challenge by developing a high-quality plant-phenotype relationship (PPR) corpus. The corpus contains information derived from over 20,000 PubMed abstracts corresponding to 5,668 plant and 11,282 phenotype entities, and demonstrates a total of 18,709 relationships.

By using GPT to summarize articles from PubMed, we hope to demonstrate the feasibility of this approach and contribute to the development of useful tools for researchers in the field. Overall, our project aims to provide a valuable resource for researchers in the bioinformatics industry, enabling them to extract meaningful information from unstructured textual data with greater ease and efficiency.

The plant-phenotype relationship (PPR) corpus was developed by Cho et al. and described in their paper "Plant phenotype relationship corpus for biomedical relationships between plants and phenotypes" (Sci Data 9, 235, 2022).

B. Related Works

The primary tool that was found to perform a similar function to ours is Textpresso Central, a customizable platform for searching, text mining, viewing, and curating biomedical literature. The tool was created to address the difficulties of knowledge retrieval and extraction given the rapid pace that biomedical literature grows at. Biomedical researchers have faced many challenges with parsing through the large volume of information revealed to find relevant details, necessitating the assistance of natural language processing and text mining. Thus, Textpresso was developed as an automated information extraction system which can mine text of biomedical journal articles from PubMed for relevant information. The tool achieves this by splitting article text into sentences and labeling these sentences with different tags. The tags are subsequently sorted in semantically meaningful categories which are defined in a shallow ontology to increase query precision. Searching using Textpresso can be performed by entering words or phrases, selecting categories from a cascading menu, or combining keywords and categories. Search results are presented as lists of sentences which users could choose to sort by relevance or by position within documents.

C. Problem Definition

The complexity and technical language of lengthy papers on plant genome research can make it difficult for researchers to read and understand. For researchers who are not specialists in the field, the amount of material in these articles—which frequently includes gene sequences, functional annotations, and genetic markers—can be overwhelming.

Researchers may not have the time or resources to read every article in detail due to the exponential growth of scientific literature, making it difficult to remain current on the most recent advancements in the field. Missed opportunities and information gaps may result from this.

By giving a succinct summary of the key conclusions of a lengthy article and emphasizing the important points, an article summarization search engine tool can assist in resolving these issues.

Biomedical researchers at Phoenix Bioinformatics have struggled with parsing through the high volume of articles containing information about genes to find text relevant to the specific genes. Because of this, our tool aims to use Natural Language Processing to identify articles that are most relevant to the user's search query.

II. Methodology

1) The first step in our design process is to present the data. This involves identifying the key information that researchers are looking for in articles related to plant genome research. In our case, this information includes gene sequences, functional annotations, and genetic markers. We use natural language processing techniques to identify and extract this information from the articles. This process involves analyzing the articles to identify key phrases and terms that are relevant to the user's search query. Once we have identified these phrases and terms, we can use them to generate a summary of the article that highlights the most important points.
2) The second step in our design process is to describe the proposed solution. Our proposed design process/solution is in tune with the work mentioned in our related works section, which includes various techniques for summarizing scientific articles. The one which ended up being implemented was a text chunking method by first summarizing small sections of the article, then summarizing those summaries into a higher-level summary, and so on. This involves identifying the most important sentences and phrases in the article and using them to generate a concise summary that highlights the key points.
3) Third, how we evaluated our proposed solution:
   We use a combination of manual and automated techniques to evaluate the quality of the summaries generated by our tool.
   We evaluate the quality of the summaries using metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which measures the overlap between the summary generated by our tool and the original article. We also conduct user studies to evaluate the usefulness of the summaries for researchers in the field.

III. Results

The results for the 2 search engine methods we used were fairly accurate, however nowhere close to what we will be presenting the client with. The search mechanism was built using a TF-IDF (term frequency-inverse document frequency) information retrieval technique that is used to search through a dataset and find the best match to a related string. This technique works by assigning a weight to each term in a document based on its frequency of occurrence in the document and the inverse frequency of occurrence in the entire dataset. The greatest precision was achieved using this method on the "Result" section of each article as using this method on every article to find a similarity would take up a lot of memory storage. There was a tradeoff between the time and space complexity of the model in this instance.

TABLE I
TABLE SHOWING THE TWO MODELS USED FOR SEARCHING THE DATASET OF ARTICLES FOR THE BEST MATCH

Model	Accuracy	Precision
MNB	76.83%	82%
TF-IDF	74.2%	88.45%

IV. Conclusion

In conclusion, our project aimed to address the complexity and technical language of lengthy papers on plant genome research by creating an article summarization search engine tool. Through the use of Natural Language Processing and TF-IDF weighting, our tool can identify articles that are most relevant to the user's search query, summarizing key conclusions and emphasizing important points.

While we believe that our tool has the potential to greatly assist biomedical researchers in their search for relevant information, there are still challenges that remain. One of the challenges is improving the accuracy of the tool in identifying relevant articles, as well as improving the quality of the summarization. Another challenge is expanding the tool to incorporate additional datasets and sources of information.

Moving forward, the next steps in the development of our project would be to continue refining the tool by incorporating user feedback and improving the algorithms used for article selection and summarization. In addition, we would focus on expanding the tool to incorporate more advanced features such as the ability to analyze gene sequences and identify potential correlations with other genes.

We believe, that by creating a universal solution to a long text summarizing tool, especially with complex and technical language, we would be able to incorporate bigger and better Large Language Models (LLMs) to make our summarizations even better. The possibility of creating a specialized, self-training Language Model purely for medically complex terms would also be viable given the perfect methodology to chunk pieces of texts into smaller pieces, all while maintaining as much context as possible.

In our opinion, the most important aspect of the project to work on next would be improving the accuracy and quality of the tool. By addressing these challenges, we believe that our tool can become a valuable resource for researchers in the field of plant genome research, enabling them to more easily access and understand the latest research findings.

References

[1] https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2103-8
[2] https://arxiv.org/abs/2010.12495
[3] https://openai.com/research/summarizing-books
[4] https://towardsdatascience.com/how-to-apply-transformers-to-any-length-of-text-a5601410af7f
[5] https://medium.com/analytics-vidhya/build-your-semantic-document-search-engine-with-tf-idf-and-google-use-c836bf5f27fb

Hazelnut: Building an intuitive AI assistant

Braulio Antonio
Queen's University
19bfac@queensu.ca

Jack Taylor
Queen's University
17jmt5@queensu.ca

Luke Major
Queen's University
luke.major@queensu.ca

Matthew Girard
Queen's University
matthew.girard@queensu.ca

***Abstract*—Natural Language Processing (NLP) and Generative Artificial Intelligence (AI) have the potential to revolutionize the way people interact with computers. However, despite these advancements, many software solutions still have a steep learning curves, or do not have an intuitive usability, making them non-appealing or challenging for a broad demographic of users to navigate and use them effectively. Hazelnut is looking to serve as a proof-of-concept of an AI powered assistant that can be fully used in a conversational way, with features that are not restricted to only text generation.**

I. Introduction

Almost every task we perform in our daily lives (at the personal or at the work level) involves handling information, particularly in the form of retrieving information (i.e. searching for information) or generating it (e.g. writing). For over 40 years, personal computers have assisted us in these two fundamental tasks. With continuous technological advances in internet communication, personal storage and cloud storage, the average amount of digital data a person deals with is 146 GB a day (including personal files and internet data exchange)or more than 50 K GB in a year.

Services like Google and Bing, which now feel ubiquitous allow us to navigate in a structured way the vast amount of information accessible through the world wide web. However, the realm of personal information does not have yet a reliable staple search engine. Possible reasons include: 1) the size of personal data is negligible compared to the size of internet data, 2) each operating system may implement tools and shortcuts for people to navigate and organize their files to their own preference and 3) intuitive user interfaces are only started to become plausible thanks to the advancement of pre-trained natural language machine learning models.

Hazelnut seeks to fill-in the gap of information management at the personal level by combining information retrieving and information generation in a single app.

It is worth noting that there are mainstream personal knowledge search engines, one of the most popular being Notion. The distinction that we make between Notion and our app, Hazelnut, is that Hazelnut will not rely on the users to input information in specific ways (through the proprietary Notion web interface) but rather through the native information processing software from the users OS, such as Microsoft word, PowerPoint, text files, etc., lifting the burden of updating, maintenance and the use of third-party web-based apps.

II. Hazelnut Architecture

The current version of Hazelnut consists on a back-end with four features: search engine, (text-containing) files organizer, contact information retrieving and chatGPT. The first 3 features correspond to information retrieving, and the generative model, to information generation. Combining intuitive tools for information retrieving and information generation leads to what we call an AI assistant.

A. Natural Language Processing

All of the current features of Hazelnut are built around Natural Language Processing (NLP), which refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can [1].

In the early days, many language-processing systems were designed by symbolic methods, i.e., the hand-coding of a set of rules, coupled with a dictionary lookup, such as by writing grammars or devising heuristic rules for stemming. More recent systems based on machine-learning algorithms have many advantages over hand-produced rules.

B. Transformers

In Machine Learning and AI, NLP consists on two closely related goals: 1) converting strings to vectors (since virtually every machine learning model works in vector spaces) and 2) maintaining or devising the meaning or relationship of words, sentences or complete paragraphs.

In recent years both goals have been achieved by transformers. A transformer is a deep learning model that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. Like recurrent neural networks (RNNs), transformers are designed to process sequential input data, such as natural language, with applications towards tasks such as translation and text summarization. However, unlike RNNs, transformers process the entire input all at once. The attention mechanism provides context for any position in the input sequence.

Transformers typically undergo self-supervised learning involving unsupervised pretraining followed by supervised fine-tuning. Pretraining is typically done on a larger dataset than fine-tuning, due to the limited availability of labeled training data. [2]

In Hazelnut we used two type of pretrained transformers: the 384 dimensional base pre-trained all-MiniLM-L6-v2 and the

768 dimensional base all-mpnet-base-v2 [3], [4]. We used the all-Mini for embedding user prompts, sentences and queries and the all-mpnet for embedding full paragraphs and full texts. Due to time constrains and limited availability of data sets we were not able to perform any fine-tuning at this time.

C. Hazelnut features

1) Semantic search: The central feature of Hazelnut is the Semantic Search, which as opposed to keyword matching, semantic search attempts to generate the most accurate results possible by understanding based on searcher intent, query context, and the relationship between words. In our case, we rely entirely on the relationship between words which is provided by the pretrained transformers [**?**].

Given two embedded vectors, e.g. an embedded prompt coming from the user and an embedded sentence from a file, we can compute how similar the two sentences are by computing the cosine of the angle between the two corresponding embedded vectors. This is what is called cosine similarity and it essentially measures the projection of one vector on to the other [**?**].

Computing the cosine similarity of a given prompt with a all the sentences inside a knowledge base allow us to implement a form of semantic search. For example, if a user asked Hazelnut "Find a file containing the periodic table", the sentence would be vectorized, and then compared with the vectorized files on their computer, and likely any sentence containing the words "periodic" and "table" would return the highest cosine similarity but also any sentence that is related to chemistry.

2) Clustering: Hazelnut enables file access and organization by categories, topics, or specifications through clustering, which groups data sets based on mathematical similarities (often Euclidean distances) using either Hierarchical or Partitioning clustering methods.

Hierarchical clustering starts by treating each point as a distinct group and then combines the two closest groups into new ones while maintaining the original groups. The process is repeated until the desired number of clusters is reached, allowing for varying levels of similarity and flexibility. It's ideal for small data sets.

Partitioning clustering splits data points into a set number of groups, then optimizes the cluster locations and their assigned data points. It creates non-overlapping collections ideal for large data sets, specified outcomes, and subsequent analysis.

Hazelnut utilizes K-Means partitioning clustering, randomly selecting "k" data points as the centroids for each cluster. Data points are then assigned to the closest centroid's group. Finally, the centroids are re-assigned to the mean of the data points in each cluster. This process repeats until the outcomes stabilize.

Hazelnut uses the Silhouette Method, which evaluates the silhouette width, the difference between the average intra-cluster distance and average inter-cluster distance produced by each "k" value in a range to determine the optimal "k" value.

3) ChatGPT: Hazelnut integrates ChatGPT, powered by Open AI's gpt-3.5-turbo model accessed via API, to enable full functionality in a user's search directory.

APIs (Application Programming Interfaces) allow different software applications to communicate by establishing a collection of functions that can be called by one application to access specified features and information from another.

Despite its power, the OpenAI API has limits, especially in the amount and speed of data requests. Language models read text in tokens. In English, a token can be as short as one character or as long as one word. OpenAI charges $0.002 per 1K for gpt-3.5-turbo. Free OpenAI accounts give users an $18 credit for up to 9 million tokens; a single call can only access 4096 tokens. These limits, while generous, could still pose issues for prolonged use.

III. Implementation

Since the most code intensive features of Hazelnut consist on a search engine, one of the pivotal components of our code is the automated creation of a knowledge base. We restricted the content of our knowledge-base to metadata (file names and locations) data (text within our files) sentences, keywords and contacts within our files, all of them stored in pandas data frames.

A. Creation of the knowledge base

The main data frame we built is the meta-data data-frame which contains file names, sub-directories and their complete file paths within a specified directory. From this data frame we run different functions to extract text from Microsoft Word documents and PowerPoint files, PDF's and text files. After extracting the text we tokenize the text per files into sentences and keywords and we save them into different data frames for easier access.

The contacts data frame is built manually, since in a realistic implementation they contact information would be provided by the user upon signing up in the app.

Every data frame (text, sentences, keywords and contacts) contains a column with its corresponding embedded vector.

B. The hazelnut API: The 'prompt' pipeline

As you can see from figure 1, prompt classification is the vital first step in deciding what functionalities of Hazelnut to utilize for a given user prompt. The prompt classifier needs to predict with high accuracy which of the four classes of Hazelnuts functionalities to access. To do this we needed to develop our own classifier model.

We began by generating an initial dataset of 25 example prompts for each category, meaning our final dataset was only around 100 prompts and their corresponding class. Within each prompt we tried to include 1-3 keywords in square brackets that could later be replaced to create a much larger augmented dataset.

We then created a dictionary containing keys matching the keywords included in the prompts and an array of values

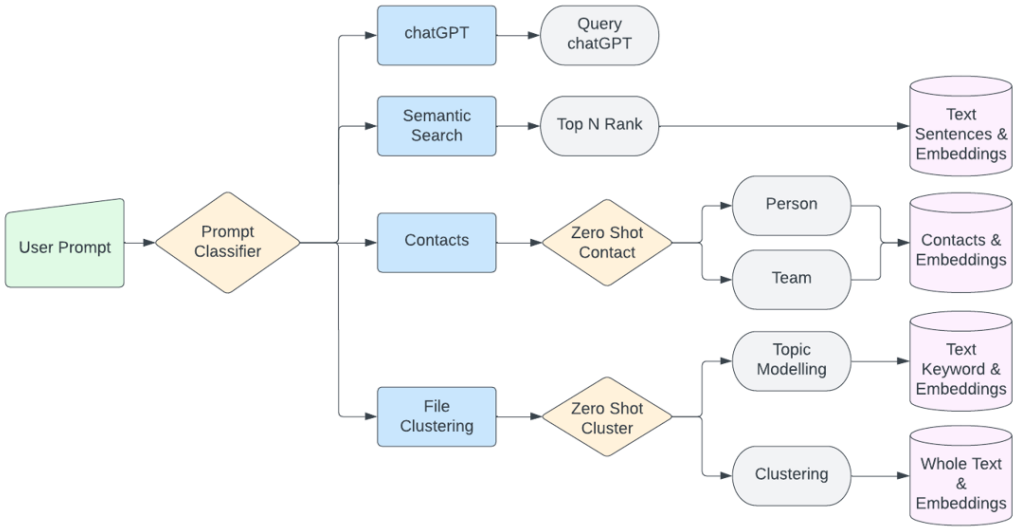

Fig. 1. Example of an image spanning the entire width.

representing possible words in that keywords category. Using this keyword dictionary, we were able to create a final augmented dataset that was approximately 10x the original size by generating prompts with all possible keyword value combinations. This gave our models a much better chance of generalizing on the overall structure of our queries as opposed to overfitting on specific examples.

Using this dataset we trained 3 separate models, a CNN, Random Forest, and State Vector Machine, and took the geometric average of their predictions for our final class prediction. Taking the geometric average of the 3 models yielded an overall better accuracy as each model was small and they could not fully cover all possible cases alone. The final accuracy was 98% on our test set.

IV. Conclusions and Potential Impact

The rapid progress in AI technology, particularly in the field of predictive text transformers, has opened new possibilities for computer interaction and capabilities. By combining these advances with other areas of AI research and traditional computing, such as clustering, embedding based data representation, and semantic search, we can create systems that redefine the traditional user interface into something more conversational and tailored to individual needs.

Hazelnut represents a first step in this direction, pushing the boundaries of how we view the systems we use daily into something more dynamic and fluid. The conversational nature of Hazelnut lowers the barrier of entry for users, enabling anyone to voice their intentions and receive personalized assistance.

The potential impact of Hazelnut and similar technologies on personal and professional productivity and organization is immense. Hazelnut's fast turnaround time on tedious tasks such as file management and business communications mean greater efficiency and engagement on previously daunting projects. As a personal assistant that maintains an extensive set of databases related to the user's personal and professional life, Hazelnut eliminates the tedium of maintaining records of conversations, contacts, and files.

Despite these benefits, machine learning carries risks of non-deterministic behavior. As users, we are accustomed to computers being precise and rigid in their operation. Given the unsupervised nature of AI training, conversational AI systems cannot guarantee that everything has been accounted for in the way that traditional programs can. Hazelnut's personalized databases help minimize this risk, but users must exercise caution to proofread and fact-check Hazelnut's work to avoid overlooking errors or omissions.

In conclusion, the potential impact of Hazelnut and other conversational AI assistants is significant, making it a promising area for further research and development. While it is important to be mindful of the risks associated with machine learning, Hazelnut represents a crucial step towards creating more personalized and intuitive interfaces that can adapt to individual needs and preferences. With continued progress in AI technology, we can look forward to a future where interacting with computers is more natural, productive, and enjoyable.

References

[1] What is Natural Language Processing https://www.ibm.com/topics/natural-language-processing#:~:text=the%20next%20step-,What%20is%20natural%20language%20processing%3F,same%20way%20human%20beings%20can.
[2] Transformers (machine learning model) https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)
[3] all-MiniLM https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[4] all-mpnet https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[5] Semantic search https://en.wikipedia.org/wiki/Semantic_search
[6] Jiawei Han, Micheline Kamber, Jian Pei, Getting to Know Your Data, Data Mining (Third Edition), Morgan Kaufmann, 2012, Pages 39-82, ISBN 9780123814791, https://doi.org/10.1016/B978-0-12-381479-1.00002-2

Lung Cancer Detection on Chest X-rays using Deep Convolutional Neural Networks

Jackson Kehoe
Queen's University
jkehoe00@gmail.com

John Zhou
Queen's University
johnzhou7913@gmail.com

David Nguyen
Queen's University
david.nguyen@queensu.ca

Frank Siyung Cho
Queen's University
20fsc@queensu.ca

***Abstract*—Early diagnosis of lung cancer is critical for early intervention and prognosis in cancer treatment. Chest X-rays are an accessible, cost-effective and low-radiation imaging method to investigate lung cancer. The team explored the application of machine learning algorithms to X-ray images to identify lung cancer. Our team used the a subset of the ChestXray-8 dataset and narrowed the data down to masses and nodules found in the lungs. The data was then split for an even distribution of cancer and non-cancer images. An AlexNet, a VGG-16 and a ResNet50 are the machine learning models that are investigated. A smaller subset of the data provided has bounding boxes, the bounding boxes will be included in the model it achieves small loss amounts. The team was able to achieve a 73.26% accuracy, a precision of 0.73, a recall score of 1.00, and an F1 score of 0.85 on the testing data. After training, the bounding box model achieved a 8.97 X e-04 loss. Further work on the project includes exploring different models to improve the scores, obtaining more bounding box data to create a localization feature, and creating an interactive application that can be used in a medical setting.**

I. Introduction

This is the paper for the QMIND team exploring machine learning algorithm applications to detect lung cancer. This paper contains all the information achieved through the design process and findings of the team.

A. Motivation

Lung cancer is the leading cause of cancer mortality in Canada and worldwide [1]-[2]. It is estimated that 30,000 were diagnosed with lung cancer in 2022, and a further 20,700 will die of it, representing approximately 24% of cancer deaths that year [3]. The Canadian health care system has been weakened due to the COVID-19 pandemic, with many health care providers facing burnout due to the stress of working overtime hours while being understaffed. In a survey of health care workers conducted in 2021, 83% of physicians said they felt more stressed, 65% stated they had an increase of workload, and 46% have had to work additional hours [4]. Due to this environment, 11% of doctors intend to leave health care or change careers within the next 3 years [4]. This loss in practitioners could further strain the system.

Wait times for medical imaging have significantly increased since the COVID-19 pandemic, with some provinces reporting median wait times of 7 to 8 weeks for a computed tomography (CT) scan and 12 to 20 weeks for a magnetic resonance imaging (MRI) scan compared to the recommended 30-day wait time [5].

Prevention, screening and early diagnosis are critical for reducing the burden of cancer and improving patient prognosis and survival rates [1]. While CT scans have been proven to be superior to chest X-rays for lung cancer screening, chest X-rays remain more accessible, cheaper and provide low exposure to radiation [6]. The highest quality studies in the literature suggest that the sensitivity of radiologist interpretation of chest X-rays for symptomatic lung cancer is approximately 77-80% [7]. Thus, augmentation of diagnostic accuracy using computer-aided detection software on chest X-rays may be warranted to further improve rates of early lung cancer diagnosis where low-dose CT scans are unavailable. These issues can be alleviated by leveraging deep convolutional neural networks (DCNNs) trained on chest X-rays to identify cancerous masses and nodules. Computer-aided detection systems using DCNNs can accelerate and assist medical providers during the diagnostic process and reduce the number of cancer cases that go undiagnosed.

B. Problem Definition

The problem is to correctly identify possible cancerous areas on the lungs based on chest radiographs and, if possible with a sufficiently high degree of accuracy, to additionally localize the cancerous tissue using bounding boxes. By classifying and localizing the presence of lung cancer on chest X-rays, the diagnostic process and accuracy of lung cancer detection can be improved. With the growing role of computer vision in radiology, we acknowledge the importance of ethical practices regarding data privacy, algorithmic biases, as well as safety and transparency.

II. Methodology

A. Data

The data used is from the Chest-Xray8 data set that is publicly available online [9]. The data set contains 108,948 frontal view X-ray images from 32,717 unique patients that have been de-identified by name and given subject IDs. It includes the subject's age, but contains no description of their lifestyle, medical history or other personal characteristics. The data has labels associated with each image for different lung and heart afflictions. The data was first tested using all lung-related labels, and subsequently narrowed down to explore the accuracy of DCNNs for classifying the presence or absence lung cancer. The labels of masses and nodules were the best

clinical indicators for predicting lung cancer on chest X-rays. The provided data had already been split into training, testing and validation data. Initially the data had approximately 1,000 cancer positive images with nearly 50,000 cancer negative images. This resulted in the model predicting negative every time. The data was further processed for an even dispersion of data. Shown below in Figure 1 is a graph showing the dispersion between cancerous and non-cancerous images.

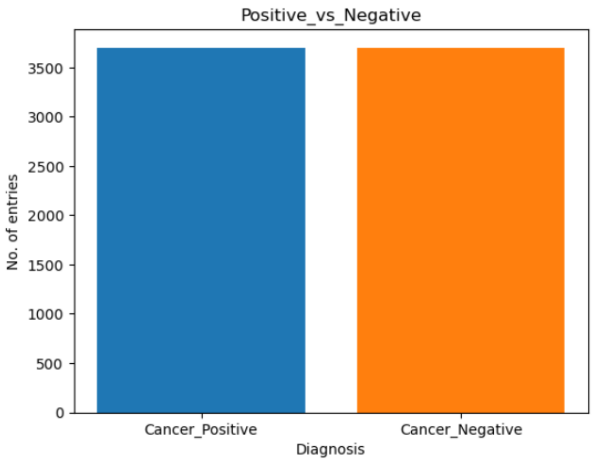

Fig. 1. Dispersion of the data used.

The images used for training and validation had a scale, shear, zoom, rotation, brightness range, horizontal and vertical flip applied to it. For testing, only a scale was applied. From these images, 6,300 images were used for the training set, 44 for the validation set, and 1,969 for the testing set.

B. Proposed Solutions

The proposed solution was to explore the accuracy of 3 different neural networks in predicting lung cancer. The chosen models were AlexNet, ResNet50 and a VGG-16 model due to their high accuracy in other machine vision applications. Each model will use the adaptive moment estimation (Adam) as their optimizer with a learning rate of 1e-4.

The AlexNet is based off a convolution neural network with 5 dense layers, 3 max pooling layers, 2 normalization layers, 2 fully connected layers and 1 softmax layer. The architecture can be seen in Figure 2.

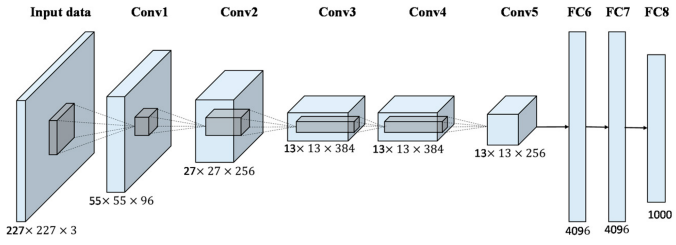

Fig. 2. Architecture of AlexNet[11].

The ResNet50 is a ResNet model consisting of 48 convolution layers with a MaxPool and Average Pool layer for a total of 50 layers. It uses ImageNet weights, which are initialized values for the layers to base their image classification on. The architecture can be seen in Figure 3.

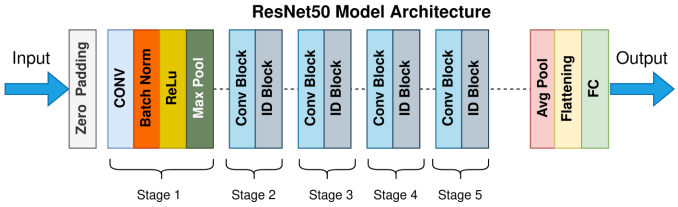

Fig. 3. Architecture of ResNet50[12].

The VGG-16 is an improvement to the AlexNet model. It has 13 convolutional layers and 3 fully connected layers, doubling the layers when compared with AlexNet. The architecture can be seen in Figure 4.

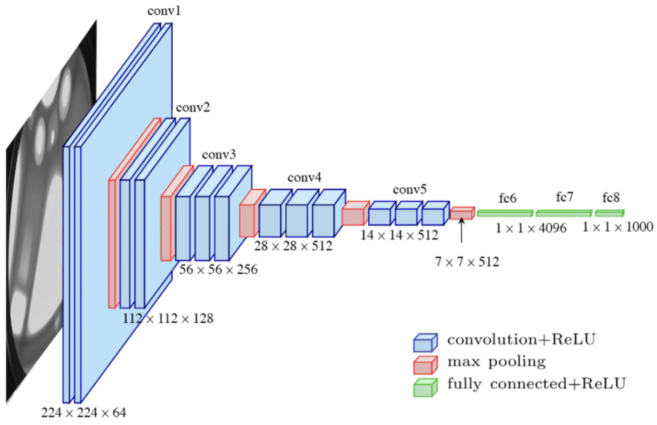

Fig. 4. Architecture of VGG-16[13].

C. Evaluation

The models are evaluated based on their accuracy when predicting positive and negative instances of cancer. 3 different accuracy measures are collected for the training, validation and testing. Additionally, measures for the precision, recall and F1-score will be collected to evaluate the effectiveness of each model. The equations for the accuracy measures are shown respectively below.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

D. Bounding Boxes

Bounding boxes will be applied to the model if a regression model is able to accurately predict the area of concern. The data for bounding boxes is limited for the X-rays with approximately 160 entries for mass and nodules annotated. Additionally, applying bounding boxes to X-rays is difficult as it does not provide perfectly clear images when compared to CT scans. Shown below in Figure 5 is the annotated bounding box on an X-ray image with a mass/nodule.

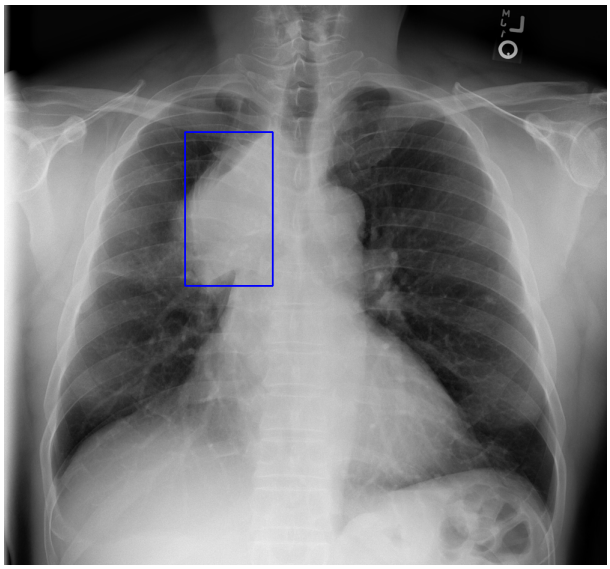

Fig. 5. Bounding Box on Training Image.

III. Results

A. Classification

The performance of the 3 models were decided based on 4 metrics: Accuracy, Precision, Recall and F1. These 4 metrics were decided to be the most relevant however, other metrics such as Specificity or Negative Predictive Value (NPV) could have been used.

The accuracy metric was used to determine how many images the model could correctly classify as having masses/nodules from the total amount of sample images; this is demonstrated below in Equation 1.

The precision metric was used to determine the number of samples which were predicted to have masses/nodules from the total amount of samples with masses/nodules; this is demonstrated in Equation 2.

The recall metric also known as sensitivity or true positive rate was used to determine how many predicted images and actual images had masses/nodules from the total amount of images with masses/nodules; this is demonstrated in Equation 3.

Finally, the F1 metric was used to determine the number of correct predictions the model made; this is demonstrated in Equation 4.

The results for the classification task for all models can be found in Table 1.

TABLE I
METRICS FOR BINARY CLASSIFICATION USING DIFFERENT MODELS

Model	Testing Accuracy	Precision	Recall	F1
VGG-16	73.26%	0.73	1.00	0.85
ResNet50	57.29%	0.50	0.85	0.62
AlexNet	49.21%	0.49	1.00	0.66

Out of the models we tested for binary classification, VGG-16 performed the best overall. VGG-16 achieved an accuracy of 73.26%, precision of 0.73 and F-1 score of 0.85 for predicting cancer versus no cancer on the testing set.

Our results could have been limited by the data set. There was a low distribution of data and low amount of data available for cancer positive versus negative. With the 50/50 distribution we utilized to disperse data, the models were not exposed to the entire range of X-rays for training and were thus not generalizable. Moreover, further limitation of this model is that the Chest-Xray8 dataset used auto-annotated labels and were estimated to only be 90% accurate, potentially contributing to inaccuracies [9]-[10]. The model could be correct in its assessment, but the labelling on the data set may have been wrong.

It must also be noted that authors of the Chest-Xray8 data set tested various models on multi-label classification had significantly lower accuracies (56.44% and 71.64% respectively) for mass and nodule classification compared to other diseases [9].

The training results of the models can be seen through figures 6 to 8. 20-30 epochs of the training and validation sets were used for training. All 3 differed in accuracies for training and validation. It is important to reiterate that the validation set only had 44 images, so interpreting the results should mainly be focused on the training accuracy.

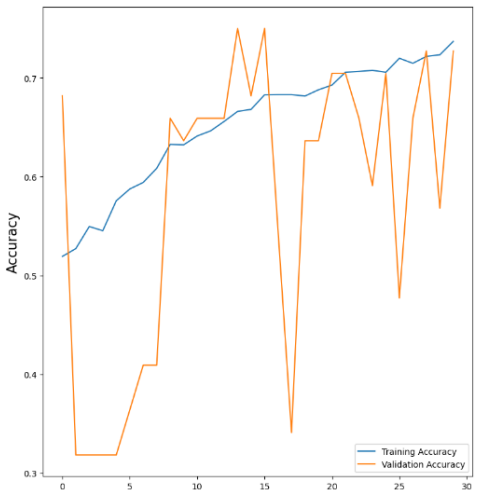

Fig. 6. ResNet50 Training and Validation Accuracy

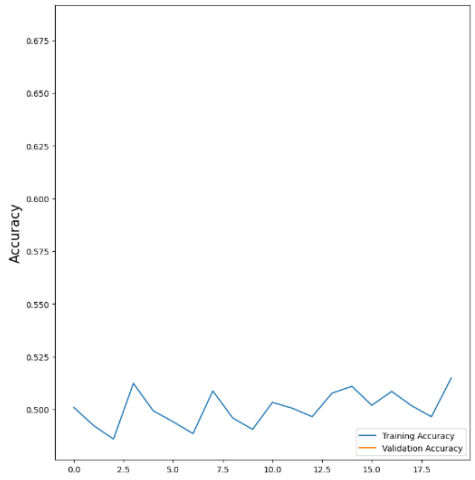

Fig. 7. AlexNet Training and Validation Accuracy

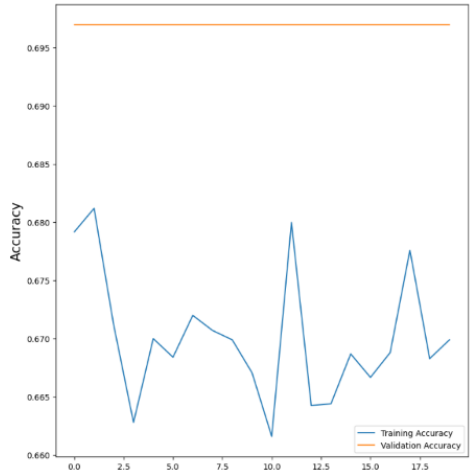

Fig. 8. VGG-16 Training and Validation Accuracy

ResNet50 gradually improves its training accuracy from about 50% to about 72%, but its validation accuracy remains inconsistent. As the accuracy has not plateaued by the end of training, this indicates that the accuracy could further improve with more epochs. AlexNet training accuracy remains steady around 50% throughout training. Due to error in running, the validation was not tracked, but the results suggest that it would be consistent with VGG-16's validation accuracy. VGG-16 produces consistent results throughout training, retraining a training accuracy around 67% and a validation accuracy around 70%. The AlexNet and VGG-16 consistent results suggest that the model have been overfitted early on and were not able to improve. This suggests that the models are too simple and could benefit from more layers, a different optimizer, a different loss metric, and/or a smaller learning rate.

B. Localization

The bounding boxes were trained on a basic CNN and loss was measured. The loss by the end of training was 8.97 X e-04 on the training data and 0.0362 on the validation data. The values on the bounding boxes were normalized before being passed into the model. So these values are considered to be large. Good results were unable to be achieved in the bounding boxes due to lack of data available and distinction within the X-ray images. The results of a sample bounding box prediction is shown in blue on the image with green representing the true bounding box in Figure 9.

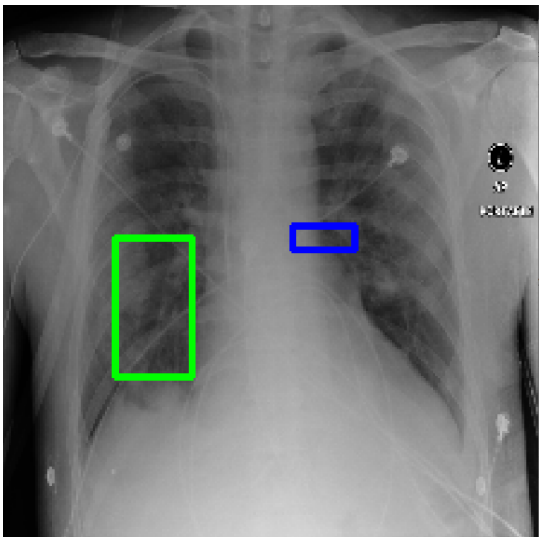

Fig. 9. Predicted bounding box shown in blue, actual bounding box shown in green.

IV. Related Works

In 2018, Ausawalaithong et al. used a DenseNet model with ImageNet weights and 121 layers to classify lung cancer through chest X-rays [14]. They trained their model on the ChestX-ray14 data set. They differed by training their model on 108 899 images with 4992 nodule images being the positives and deemed cancerous, but kept a 50/50 split of cancer non-cancer for the validation and testing set. They did not include masses as positives. 108 899 images were used for training, 2048 images were used for validation, and 532 images were used for testing. The models were trained on an unspecified number of epochs. On the data set, they achieved an accuracy of 84.02%, a specificity of 85.34%, and a recall value of 82.71% for nodule detection.

In this paper, X-ray scans were the primary medical imaging procedure used to identify masses/nodules. However, other imaging techniques, such as PET scans or MRI scans, can be used in order to obtain accurate results when applying machine learning models. In 2021, Idrahim et al. used a data set comprised of 33 676 X-ray and CT scans to 4 different image classification models, a VGG19-CNN, a ResNet152V2, a ResNet152V2 + Gated Recurrent Unit (GRU), and ResNet152V2 + Bidirectional GRU (Bi-GRU) to identify COVID-19, pneumonia, and lung cancer [15]. Of the 4 techniques, the VGG-19 model was deemed to have the most accurate results with an accuracy of 98.05%, a precision of 98.43%, a recall of 98.05% and a F1 score of 98.24%.

V. Conclusion

In this paper, 3 convolutional neural networks, an AlexNet, a ResNet50, and a VGG-16, were trained on chest X-ray images from the Chest-Xray8 data set. After applying data augmentation on an even split of images showing cancer and no cancer and splitting the data into training, validation, and testing sets, these models were unable to produce sufficiently high accuracy, precision, recall, and F1 scores for classifying chest X-ray images for lung cancer. Of the 3, VGG-16 performed the best with an accuracy of 73.26%, a precision of 0.73, a recall of 1.00, and an F1 score of 0.85. The bounding box regression model was also unable to produce bounding boxes that accurately mapped the area with masses or nodules.

A. Future Work

As evident in the DenseNet model produced by Ausawalaithong et al. on the same data set used in this paper, high scores in classifications are possible. This suggests a number of changes that could improve results: (1) Add more layers in the network - the DenseNet used 121 layers, while the ResNet50, the largest model evaluated in this paper, had 50 layers. (2) Add more images to validation - the DenseNet had 2048 images for validation while the 3 models had 44 images to validate from. Validation is not used to train the model, rather, it provides insight on how it performs while training. With a clearer output, researchers will have a better understanding on how to improve it. (3) Run for more epochs - more epochs should help the models converge in accuracy. As seen in the ResNet50 accuracy plot, the model has large spikes throughout training and had not plateaued in its learning yet. (4) Use a different optimizer, loss function, and/or decrease the learning rate - the AlexNet and VGG-16 models converged early on and do not suggest improvement through more epochs. Changing 1 or all 3 could improve on this.

The same suggestions can be applied to the bounding box regression model, however this can also be attributed to the small size of the data set with exact positions of mass. More lung cancer chest X-ray data sets should be considered to supplement this. In addition, since Chest-Xray8 had the limitation of using auto-annotated labels, further steps include using radiologist-annotated datasets. Employing a transfer learning approach with these DCNNs may also significantly improve performance with a low number of epochs. Furthermore, novel approaches employing an ensemble methodology with multiple DCNNs may improve the accuracy of predictions for lung cancer and diagnosis using chest X-ray images [16].

References

[1] R. Nooreldeen and H. Bach, "Current and Future Development in Lung Cancer Diagnosis," *IJMS*, vol. 22, no. 16, p. 8661, Aug. 2021, doi: 10.3390/ijms22168661.

[2] J. Ferlay et al., "Cancer incidence and mortality worldwide: Sources, methods and major patterns in GLOBOCAN 2012: Globocan 2012," *Int. J. Cancer*, vol. 136, no. 5, pp. E359–E386, Mar. 2015, doi: 10.1002/ijc.29210.

[3] D. R. Brenner et al., "Projected estimates of cancer in Canada in 2022," *CMAJ*, vol. 194, no. 17, pp. E601–E607, May 2022, doi: 10.1503/cmaj.212097.

[4] S. C. Government Of Canada, "Impacts experienced by health care workers during the COVID-19 pandemic, by occupation, Canada, September to November 2021," Jun. 03, 2022. https://www150.statcan.gc.ca/n1/daily-quotidien/220603/cg-a001-eng.htm (accessed Mar. 13, 2023).

[5] Fraser Institute, "Waiting Your Turn: Wait Times for Health Care in Canada, 2022 Report," 2022. [Online]. Available: https://www.fraserinstitute.org/sites/default/files/waiting-your-turn-2022.pdf

[6] E. J. van Beek, S. Mirsadraee, and J. T. Murchison, "Lung cancer screening: Computed tomography or chest radiographs?," *World J Radiol*, vol. 7, no. 8, pp. 189–193, Aug. 2015, doi: 10.4329/wjr.v7.i8.189.

[7] S. H. Bradley et al., "Sensitivity of chest X-ray for detecting lung cancer in people presenting with symptoms: a systematic review," *Br J Gen Pract*, vol. 69, no. 689, pp. e827–e835, Oct. 2019, doi: 10.3399/bjgp19X706853.

[8] M. M. Jassim and M. M. Jaber, "Systematic review for lung cancer detection and lung nodule classification: Taxonomy, challenges, and recommendation future works," *Journal of Intelligent Systems*, vol. 31, no. 1, pp. 944–964, Jan. 2022, doi: 10.1515/jisys-2022-0062.

[9] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3462–3471. doi: 10.1109/CVPR.2017.369.

[10] B. Wang et al., "Automatic creation of annotations for chest radiographs based on the positional information extracted from radiographic image reports," *Computer Methods and Programs in Biomedicine*, vol. 209, p. 106331, Sep. 2021, doi: 10.1016/j.cmpb.2021.106331.

[11] X. Han, Y. Zhong, L. Cao, and L. Zhang, "Pre-Trained AlexNet Architecture with Pyramid Pooling and Supervision for High Spatial Resolution Remote Sensing Image Scene Classification," *Remote Sensing*, vol. 9, no. 8, p. 848, Aug. 2017, doi: 10.3390/rs9080848.

[12] S. Mukherjee, "The Annotated ResNet-50," *Medium*, Aug. 18, 2022. https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758 (accessed Mar. 13, 2023).

[13] M. Ferguson, R. Ak, Y.-T. T. Lee, and K. H. Law, "Automatic localization of casting defects with convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec. 2017, pp. 1726–1735. doi: 10.1109/BigData.2017.8258115.

[14] W. Ausawalaithong, S. Marukatat, A. Thirach, and T. Wilaiprasitporn, "Automatic Lung Cancer Prediction from Chest X-ray Images Using Deep Learning Approach," in *2018 11th Biomedical Engineering International Conference (BMEiCON)*, Nov. 2018, pp. 1–5. doi: 10.1109/BMEiCON.2018.8609997.

[15] D. M. Ibrahim, N. M. Elshennawy, and A. M. Sarhan, "Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases," *Comput Biol Med*, vol. 132, p. 104348, May 2021, doi: 10.1016/j.compbiomed.2021.104348.

[16] L. Vogado, F. Araújo, P. S. Neto, J. Almeida, J. M. R. S. Tavares, and R. Veras, "A ensemble methodology for automatic classification of chest X-rays using deep learning," *Computers in Biology and Medicine*, vol. 145, p. 105442, Jun. 2022, doi: 10.1016/j.compbiomed.2022.105442.

Organic Post Prediction

Robbie Huang
Queen's University
20rh1@queensu.ca

Stuart Fong
Queen's University
stuart.fong@queensu.ca

Jacob O'Neil
Queen's University
19jmon1@queensu.ca

Brian Grigore
Queen's University
20bg1@queensu.ca

Flora Lin
Queen's University
h.flora.lin@gmail.com

***Abstract*—The goal of the organic post prediction is to help social media managers save time in choosing images for social media, and be able to accurately predict the potential of a post on social media. We use intrinsic image popularity to help with our research. And basing it off a Convolutional Neural Network, and optimizing by ranking the thousands of popularity-discriminable image pairs. With testing results achieving a respectable performance on Instagram.**

I. Introduction

In recent years we have witnessed a rapid increase in social media usage on platforms such as Instagram, Facebook, TikTok, etc. Instagram users post an average of 46,740 photos every minute [1], with the average engagement rate of a photo being 0.65%.[2] With trends that come and go, it is hard to keep up. The job of a social media manager is tough as they have to choose from dozens of photos to post on Instagram and hope they are able to reach the target audience with the algorithm. This paper aims to help social media managers by creating a method to try to predict popularity on social media. We use the method of popularity-discriminable image pairs (PDIPs), to help reduce the influence of other factors that may cause the popularity of a photo. Using this idea we used over 66 000 images that were scraped off Instagram to use as our data set for the model. We then use a Convolutional Neural Network (CNN) to predict the popularity of images.

A. Related Works

A paper that really influenced our project was the Intrinsic Image Popularity Assessment paper written by Keyan Ding, Kede Ma, and Shiqi Wang from the City University of Hong Kong. Their approach is also using intrinsic image popularity as well as popularity-discriminable image pairs. We tried to recreate their results using the Siamese architecture as well. [3]

B. Problem Definition

The process of selecting the most engaging photo for organic social media posts is a time-consuming and costly task for social media account managers of both large brands and small businesses. This task requires a human to choose the most engaging photo from a collection of dozens of possible photos multiple times each day. We believe that there is sufficient data available online to automate this decision-making process and that a model can be developed to select the most engaging photo from a batch of photos, leading to better and faster decision-making for social media account managers. The goal of this project is to create a prototype web app that recommends the best photo to post from a user-uploaded batch of photos. The project's minimum viable product includes a generic model for recommending the best photo, which a user can choose to accept or reject.

The scope of the project focuses on the base case of helping a social media manager select the most engaging photo from a collection of photos. However, the project will include a model trained specifically on the user account's genre/theme, enabling the collection of user decisions in a database for future training. The project's success will be determined by real-world user feedback from the client, which can be used to improve the model's performance and add additional features to meet the client's requirements.

The team will need to consider audience-specific models for fine-tuning the account's specific audience/theme and defining "engagement" for the industry standard definition. The project will require a data collection and preparation component, where an existing relevant image data set can be used initially. The team will eventually need to train the model on more relevant data using prediction-based models.

Overall, the project's objective is to increase the efficiency and leverage of social media account managers by providing a tool that enables faster and better decision-making for posting on social media platforms.

II. Methodology

We collected and processed the images from Instagram as follows:

1) About 66 000 images were collected by web scraping images off of the Instagram website using the methods described by Ding et al [3]. This process included snowball sampling through the followers of each account and then removing 80% of users in order to reduce the similarities between the sampled Instagram accounts.
2) Images from the same account that were uploaded within a short period of time from each other were paired together. These images were selected to be more than one month old so that the number of likes stabilized, and had few characters in the caption in order for the number of likes to be more reliant on the uploaded

image. The image with a more significant amount of likes was assigned as the first image in the pair.

3) The images were normalized to have a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225], and then resized to 224 x 224 px to match the specifications for the pre-trained ResNet-50 model.

The model we used was a ResNet-50 network [citation] that was pre-trained on the ImageNet-1k dataset. We replaced the last fully connected layer with a fully connected layer with one output that represents the image's "popularity score." The learning rate of the ResNet-50 part of the network was set to 10^{-5}, while the learning rate of the last fully connected layer was set to 10^{-4}. The fully connected layer's parameters were initialized with the method described by He et al. [citation].

The model was trained as a siamese neural network. The images were passed into the model separately and a popularity score was obtained for each. By subtracting the first popularity score from the second, we obtain a number representing how much more popular the first image is predicted to be than the other. The sigmoid of this difference was computed and was compared to the ground truth value of 1 through the binary cross entropy loss function.

We used the Adam optimizer with an L2 penalty multiplier of 10^{-4}. The learning rates decayed by a factor of 0.95 after each epoch.

III. Results

Below is an example of a table to display results. You can add columns and rows. Describe what information is presented in the table in the caption. Make sure to describe the significance of the information presented in each table and figure. Also, here is a useful website for generating tables.

TABLE I
EXAMPLE TABLE SHOWING THE RESULTS OF AN EXPERIMENT.

Model	Accuracy	Precision	Recall	F1
CNN	97.78%	82.32%	88.66%	90.61%
SVM	86.43%	78.41%	67.43%	55.21%
RNN	79.21%	94.13%	80.03%	75.79%

IV. Conclusion

We were able to conduct the organic post prediction, using our model and the popularity-discriminable image pair and only basing popularity off the image of the Instagram post. We believe that once we are able to rank them by the image itself we can also further this project by bringing in Natural-Language-Processing techniques, such as analyzing the caption, location tag, and hashtags. Although a challenging problem with that would be to keep up with trends that appear every day on Instagram, thus having to continuously scrape data off Instagram and training the model to be able to keep up with trends, this was one of the biggest challenges for us as a team. We wanted to find a data set that would be recent enough that when training could still predict the Instagram algorithm however, all the data sets that we were able to find were all outdated by years or several months. To resolve this problem we think that a continuous scraping algorithm and training have to be done.

References

[1] Bernard Marr. (2021, July 13). How much data do we create every day? the mind-blowing stats everyone should read. Bernard Marr. Retrieved March 12, 2023, from https://bernardmarr.com/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/

[2] Marino, M. (n.d.). Average engagement rate on Instagram [updated Feb 2023]. Oberlo. Retrieved March 12, 2023, from https://www.oberlo.ca/statistics/average-engagement-rate-on-instagram

[3] Ding, K., Ma, K., amp; Wang, S. (2019). Intrinsic image popularity assessment. Proceedings of the 27th ACM International Conference on Multimedia. https://doi.org/10.1145/3343031.3351007

Predicting the NHL Draft

Ethan O'Brien
Western University
eobrien.hba2024@ivey.ca

James Hutchins
Western University
jhutch63@uwo.ca

Shaz Khan
Western University
skan746@uwo.ca

***Abstract*—This paper aims to develop a predictive model that accurately forecasts if a player will be selected in the first round of the National Hockey League (NHL) draft based on amateur performance statistics. To achieve this goal, the datasets were pre-processed by removing null values, outliers, and highly correlated variables. All categorical variables were converted to one-hot encoded variables, and columns with vast variabilities were removed to improve the quality of the dataset.**

Class imbalance was checked, and the class weights in each classification model were balanced to ensure accurate predictions. The dataset was then standardized and split into training and testing sets, and Principal Component Analysis (PCA) was used to reduce the number of features in the dataset.

Several classification models were tested, including Logistic Regression, K-Nearest Neighbours, Decision Trees, Support Vector Machine, and a Multi-Layer Perceptron Neural Network. These models were evaluated based on their accuracy, precision, recall, and F1 score.

To improve the base models, ensemble methods such as Random Forests, Bagging, and Boosting were used, and the hyperparameters were fine-tuned using a GridSearch algorithm. After evaluating the performance of different models, a bagging SVM model was selected as the final model with C equal to 100 and gamma equal to 1, achieving an accuracy of 84%.

Overall, this model provides NHL teams with valuable insights to inform their draft strategies, allowing them to make data-driven decisions and choose high-performing players while saving time and resources. By exploring various classification models, this study has demonstrated the effectiveness of machine learning algorithms in predicting NHL draft outcomes.

I. Introduction

NHL teams use substantial resources and time scouting players to determine what round to draft them. Consequently, teams must find ways to make more informed and data-driven decisions, as draft pick mistakes are costly and can directly influence the team's future success.

A. Motivation

The NHL draft is an important event for all teams, and selecting the proper player can significantly impact the team's performance. However, predicting the draft round in which a player will be selected is a difficult task that requires evaluation of various factors such as a player's skill set, physical attributes, and performance history. According to a study conducted by the NHL in 2018, NHL teams spent an average of $320,000 on amateur scouting during the 2016-2017 season [1]. However, poor draft picks don't only cause monetary costs. For example, in a 2018 interview with Sports Illustrated, former NHL general manager Craig Button stated:

> the cost of a first-round mistake can be catastrophic. A bad draft pick can set a team back for several years in terms of both on-ice performance and financial investment. [2]

Therefore, a predictive model for the NHL draft would provide teams with valuable insights to inform their draft strategies, improving their chances of selecting high-performing players while saving resources.

B. Related Works

Previous research has explored the NHL draft and attempted to predict the success of drafted players. For instance, one study utilized machine learning algorithms to predict a player's career performance based on their pre-draft statistics [3]. Another study focused on identifying the critical factors that contribute to a player's success in the NHL, such as their point production and draft position [4]. However, no research focuses specifically on predicting the round in which a player will be drafted. Therefore, this paper aims to address this gap in the literature by developing a predictive model that can accurately forecast draft rounds.

C. Problem Definition

The objective of this paper is to create a predictive model that can accurately forecast the draft round in which a player will be selected in the NHL draft. Specifically, the scope will focus on the first round as these players produce or cost organizations the most capital, as they can become superstars or a bust. To accomplish this objective, several player attributes will be analyzed, such as player position, physical measurements, and performance history, to evaluate how these fields influence draft-round selection. Ultimately, the goal is to provide teams with a valuable tool to inform their draft strategies, enabling them to choose high-performing players while saving time, and resources, and allowing the team to make data-driven decisions.

II. Methodology

A. Data Processing

The purpose of this paper was to develop a classification model to predict the likelihood of a professional hockey player being drafted in the first round based on their amateur performance statistics. To achieve this, two datasets were utilized. The first contains NHL draft round picks from the NHL API and the other performance data from the Elite Hockey Prospects website, which was obtained using a scraper.

The datasets were merged using the player's name and only the most recent amateur year's stats for each player were kept.

First, the dataset was examined for missing values and duplicates using the Pandas library in Python. Fortunately, there were only a few null values that could be removed as they did not compromise the dataset.

Next, the dataset was tested for outliers using the z-score technique provided by the Scipy stats library. The test revealed extreme values for columns P (points) and GP (games played). These rows consisted of some first-round picks with low stats and some non-first-round picks with high stats, predominantly from international leagues. External research validated that this data was incorrect and that the error resulted from misinformation on the Elite Hockey Prospect website. The outliers were removed to improve model performance.

All categorical variables, such as player position, were converted to one-hot encoded variables, and columns with vast variabilities, such as league and nationality, were removed. The collinearity of the dataset was evaluated using a correlation matrix. The analysis identified that columns A (assists), G (goals), and P were highly correlated, indicating collinearity. This issue was addressed by removing the P column from the dataset.

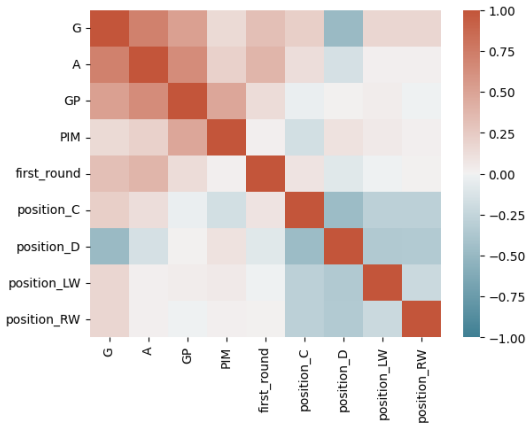

Fig. 1. Correlation Matrix.

The dataset was then checked for class imbalance. 9% of the data was first-round picks, which is consistent with the number of rounds in the NHL draft. Therefore, the dataset is unbalanced and so the class weights in each classification model must be balanced to ensure that the first-round and non-first-round picks are equally represented.

After cleaning the data, it was processed for analysis. It was divided into features (x) and target (y) variables. The features dataset was standardized using the Scikit-Learn library to ensure that the scale of the data did not affect the classification models. This involved transforming the data to a standard score by subtracting the mean and dividing it by the standard deviation. The standard scaler was saved as a joblib file so that it could be used to transform new data during the prediction stage.

Finally, the dataset was split into training and testing sets using Scikit-Learn, with a ratio of 80:20. This will allow the machine learning models to be trained on a subset of data and tested on another.

B. Feature Engineering

Principal Component Analysis (PCA) was used to reduce the number of features in the dataset while preserving as much information as possible. PCA transforms the data into a lower-dimensional space by identifying the most important features, known as principal components [5].

A PCA algorithm was employed by calculating a covariance matrix and the resulting eigenvectors and eigenvalues. These are used to rotate the n-dimensional space into a smaller space. The information proportion was calculated by dividing each eigenvalue by the sum of all eigenvalues, producing the below Scree plot which demonstrates the proportion of information explained by each principal axis. The algorithm was verified by matching the results to Scikit-Learn's PCA module. The analysis revealed that six axis', G, A, GP, PIM, position_C, and position_D, account for 85% of the data variation.

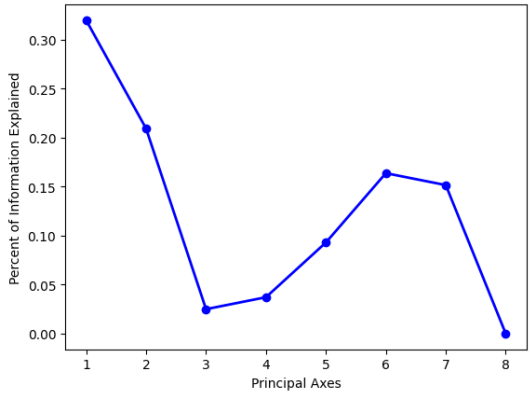

Fig. 2. Scree Plot.

In addition, wrapper methods were used to verify the best subset of features for the model. Sequential forward and backward feature selection was used with and without replacement, as well as recursive feature selection with logistic regression as the test model. Sequential forward selection (SFS) iteratively adds features to the model based on their performance until a predetermined number of features is reached, while Sequential backward selection (SBS) iteratively removes features from the model [6]. Both SFS and SBS can be used with or without replacement, depending on whether a feature that has been selected or removed is allowed to be reconsidered in subsequent iterations. Recursive feature elimination (RFE) iteratively removes the feature that has the lowest importance score. The analysis revealed that the model performed the best for the same 6 features as the PCA analysis.

Overall, feature engineering reduced the dataset's dimensionality, improving computational efficiency and allowing for more sophisticated models to be deployed given CPU restrictions on the test computer.

C. Model Selection

To determine the best classification model for the proposed problem, various machine and deep learning models were employed, such as logistic regression, k-nearest neighbors, decision trees, support vector machines, and neural networks.

The first model applied was a simple Logistic Regression model (LR). LR models use the log-odds or sigmoid function to model the connection between a group of independent variables and a binary dependent variable. The model determines the probability that an observation belongs to a specific class and then classifies the observation based on a predetermined threshold [7]. The Scikit-Learn library was used to train this model.

Next, a k-nearest neighbors (KNN) model was employed, which assumes that data points belonging to the same class will be close to each other in space. To classify a new data point, KNN calculates the distance between the new data point and all the known data points of a particular class. Based on the value of K (i.e., the number of closest neighbors to evaluate), it applies the majority rule to assign the new data point to a particular class [8]. The Scikit-Learn library was used to train this model.

In addition, a decision tree (DT) model was used. DT models function by recursively splitting the data based on the values of the features that best differentiate the data into distinct groups. Each split creates two new branches, and the algorithm continues splitting until it reaches a stopping criterion, such as a minimum number of observations in each group or a maximum tree depth [8]. The resulting tree is a hierarchical structure that can be used to classify new data points by following the branches from the root node down to the leaf node corresponding to the predicted class. The splitting criterion used was the Gini index, which measures the probability of misclassifying a randomly selected observation in a given subgroup. The max depth was set to 3 to avoid overfitting; however, this will be tuned. The Scikit-Learn library was used to train this model with the architecture in figure 3.

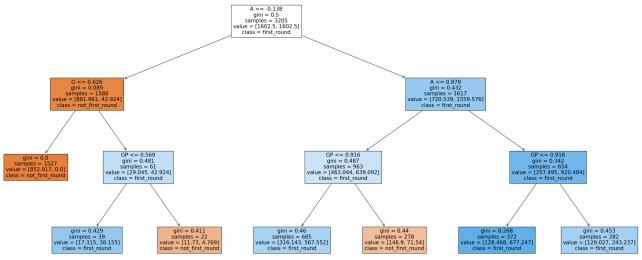

Fig. 3. Decision Tree Plot.

A support vector machine (SVM) model was used. It functions by defining a decision boundary that maximally separates the data points of different classes. The decision boundary is typically a hyperplane in a high-dimensional space that is defined by the support vectors, which are the data points closest to the decision boundary. SVM seeks to maximize the margin between the support vectors on either side of the decision boundary. In cases where the data cannot be linearly separated, SVM uses kernel functions to transform the data into a higher-dimensional space where linear separation is possible [10].

The radial basis function (RBF) was used. It operates by mapping the data into a high-dimensional space where a linear decision boundary can be defined. The RBF kernel calculates the similarity between two data points using a Gaussian distribution centered at the data point [8]. Other hyper-parameters, C, and gamma were set to defaults until tuning. The Scikit-Learn library was used.

Finally, a Multi-layer Perceptron (MLP) feedforward neural network classifier was built using the Keras and TensorFlow libraries. Each layer of the MLP model consists of a set of neurons, where each neuron takes inputs, performs a linear transformation, and then applies a non-linear activation function to the output [9]. The weights and biases of the neurons are adjusted during training to improve the accuracy of the predictions. The output of one layer serves as input to the next layer, and the process repeats until the final output is produced.

The model deployed has four hidden layers, each with 128 or 64 neurons, and the ReLU activation function is applied to the output of each hidden layer. The ReLU activation function sets all negative values to zero and preserves positive values, which allows the model to learn complex non-linear relationships between the input and output variables [10].

Batch normalization is a technique used to improve the performance and stability of deep neural networks. It normalizes the inputs of each layer by subtracting the mean and dividing by the standard deviation of the batch, which helps to reduce the effects of input covariance shift and can improve training convergence [11].

Dropout regularization is a technique used to prevent overfitting in neural networks. It randomly drops out (sets to zero) a proportion of the neurons during each training iteration, which reduces the interdependence of the neurons and can improve the generalization performance of the model [13].

The final output layer of the model has one neuron and a sigmoid activation function, which outputs a probability value between 0 and 1 that represents the predicted probability of the input belonging to the positive class. The binary cross-entropy loss function is used to measure the difference between the predicted output and the actual output, and the Adam optimizer is used to minimize the loss during training by adjusting the weights and biases of the neurons [11].

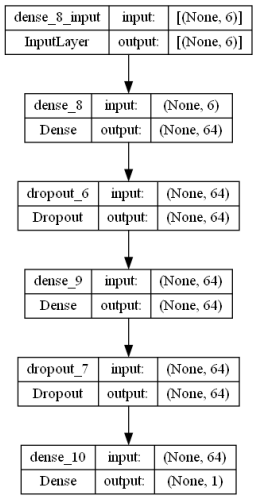

Fig. 4. Neural Network Architecture.

After training and testing each model, the performance was evaluated against accuracy, precision, recall, F1 score, and a confusion matrix.

D. Model Improvement

1) Ensemble Methods: After the base models were explored, ensemble methods were used to improve the performance. Ensemble methods were applied to the LR, DT, and SVM models.

One ensemble method that was utilized was the Random Forests method, which was applied to the DT base model. This method involves building multiple decision trees that collaborate to classify new data points. The final classification is based on the most popular classification from all the trees. Bootstrapping is used to create new datasets for each tree, which reduces the variance of the model and leads to better generalization [**?**].

The scikit-learn Bagging Classifier was also used to improve the DT, LR, and SVM models. This technique involves training multiple models on different subsets of the dataset and aggregating their predictions to form the final output. This helps to reduce over-fitting and increase the stability of the model [12]. Figure 5 provides a visualization of the bagging process [13].

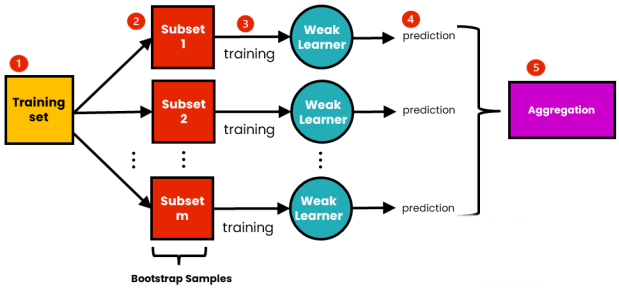

Fig. 5. The Bagging Process.

Boosting was also explored to improve performance. This is a sequential learning technique where each base model is a weak learner and builds off the previous model. In Adaptive Boosting or AdaBoost, each base model is reweighted to reduce error, and subsequent models attempt to fix the errors in the previous stage. This approach helps to improve the model's accuracy and reduces bias [12].

2) Hyper-parameter Tuning: A Grid Search algorithm was used to tune the hyper-parameters of each base model. This technique searches through a specified parameter space to find the optimal combination of hyper-parameters. The Grid Search algorithm tested each combination of specified hyper-parameters by training the model on the training data and evaluating its performance on the validation data. The combination of hyper-parameters that produced the highest accuracy was selected as the optimal combination.

III. Results

The discussed base models were trained and tested on subsets of the dataset. The model performance is as follows:

TABLE I
BASE MODEL PERFORMANCE RESULTS.

Model	Accuracy	Precision	Recall	F1
LR	77%	64%	78%	65%
KNN	88%	68%	58%	60%
DT	66%	62%	78%	58%
SVM	78%	66%	84%	68%
MLP	70%	63%	81%	61%

The KNN model performed the best with an accuracy of 88%; however, this was because the Scikit-Lean KNN Classifier does not support model weighting and so the classes were imbalanced, shown by the lowest Recal Score of 58%. Therefore, this result must be disregarded and instead the best-performing base model was the SVM with 78% accuracy. The MLP Neural Network also performed well with a Recal Score of 81% indicating high performance at classifying the under-weighted first-round pick class.

The LR, DT, and SVM models were then improved using the discussed ensemble methods.

TABLE II
TUNED MODEL PERFORMANCE RESULTS.

Model	Accuracy	Precision	Recall	F1
Random Forests	77%	64%	79%	65%
Bagging DT	69%	62%	78%	60%
Bagging SVM	82%	65%	73%	67%
Bagging LR	77%	63%	77%	65%
AdaBoost DT	67%	62%	77%	58%
AdaBoost SVM	88%	44%	50%	47%

The DT base model was improved by 11% to an accuracy of 77% using the Random Forests Method. Similarly, the Bagging SVM model was improved by 4% to an accuracy of 82%. However, this was at the expense of a decrease in the Recall score which is an important metric considering the class imbalance. The Bagging DT resulted in a slight

accuracy improvement and the Bagging LR saw no change in performance. The AdaBoost SVM model had the highest accuracy at 88%; however, it had the lowest recall at 50%. The model did not correctly classify any of the first-round picks and so must be disregarded.

Overall, the Bagging SVM model performed the best. It predicted 38 false negatives and 105 false positives. These results are expected as the stats for players drafted at the bottom of the first round compared to players drafted at the top of the second round are very similar. However, this analysis could indicate that a portion of the players drafted in the first round did not deserve to be. It would be interesting to explore if these players were draft busts or lived up to first-round expectations.

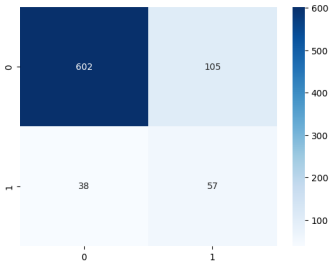

Fig. 6. Bagging SVM Confusion Matrix.

The GridSearch algorithm was used to fine-tune the hyper-parameters of the Bagging SVM and Random Forest DT classifier and further improve model performance. For the Bagging SVM, the optimal hyper-parameters were C=100 and gamma=1. For the Random Forest DT, the optimal hyper-parameters were max_depth=7 and criterion='gini'. The Bagging SVM accuracy was improved by 2% to 84% and the Recall Score by 5% to 78%.

Overall, this paper concluded that an SVM model using the Bagging technique, and hyper-parameters of 100 for C and 1 for gamma resulted in the greatest model performance. NHL teams can now predict if a player deserves to be drafted in the first round with 84% accuracy, potentially saving teams millions of dollars by avoiding draft busts and reducing scouting costs.

IV. Conclusion

The NHL draft is an essential event for all teams, and selecting the right player can significantly impact the team's performance. Poor draft picks can cause substantial costs, and therefore, a predictive model for the NHL draft would provide teams with valuable insights to inform their draft strategies, improving their chances of selecting high-performing players while saving resources.

Based on the analysis conducted, a Bagging SVM model with tuned hyper-parameters was found to be the best predictor of whether a player will be drafted in the first round of the NHL draft. The model achieved an accuracy score of 84% and a Recall score of 73%.

Moving forward, future research could explore the prediction of other rounds in the NHL draft and analyze the impact of additional variables, such as player character and work ethic, on draft-round selection. Expanding the model would greatly increase the value to NHL teams. In addition, it would be interesting to explore in more detail which players were classified as false negatives. The model could be validated by matching false negatives to draft busts and false positives to eventual superstars. It can also be tested on the upcoming NHL draft.

In conclusion, this paper addressed the gap in the literature by developing a predictive model that could accurately forecast the first round of the NHL draft. It employed a thorough Machine Learning process including data cleaning, data exploration, feature engineering, model testing, ensemble methods, and hyper-parameter tuning. It demonstrated the importance of a clean and viable dataset and how base models can be improved with additional machine-learning techniques.

References

[1] NHL. (2018). NHL teams invest more in amateur scouting and development. Retrieved from https://www.nhl.com/news/nhl-teams-invest-more-in-amateur-scouting-and-development/c-299201546

[2] Button, Craig. "The Cost of a First-Round Mistake Can Be Catastrophic." Interview with Dan Falkenheim. Sports Illustrated, 25 Apr. 2018, https://www.si.com/nhl/2018/04/25/nhl-draft-scouting-players-craig-button-interview

[3] O'Reilly, N., & Williams, P. (2019). Predicting career performance of NHL players using pre-draft statistics and physical measurements. Journal of Sports Analytics, 5(2), 95-107

[4] Larivière, J., Gauthier, A., & Lapierre, M. A. (2019). The importance of point production and draft position in predicting the success of NHL players. Journal of Quantitative Analysis in Sports, 15(3), 157-170

[5] Abdi, H., & Williams, L. J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4), 433-459. https://doi.org/10.1002/wics.101

[6] Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.

[7] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

[8] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer-Verlag.

[9] TensorFlow. (n.d.). Guide to Keras Sequential Model. Retrieved from https://www.tensorflow.org/guide/keras/sequential_model

[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press

[11] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. International Conference on Machine Learning (ICML), 448-456.

[12] Kumar, D., & Sharma, A. (2021). Ensemble Learning for Heart Disease Classification using Machine Learning Algorithms. In 2021 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[13] Analytics Vidhya. (2023, January 30). Ensemble Learning Methods: Bagging, Boosting, and Stacking. Retrieved February 27, 2023, from https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

Representations of Personality Features using Modified Autoencoders

Brandon Cheng
Queen's University
brandonkhcheng123@gmail.com

Chloe Atherton
Queen's University

Darwin Chen
Queen's University
darwinchen8@outlook.com

Molly Shillabeer
Queen's University
mollyshillabeer@gmail.com

***Abstract*—Complex systems can often be represented in lower dimensions, and doing so can often grant important insights to the system by making it easier to analyze. However, many data compression and reduction techniques face a trade-off between generalizability and accuracy. In this study, we develop a model that aims to tackle generalizability without sacrificing accuracy, and test this model by training it to learn a representation of personality features using self report survey data. Overall, results were inconclusive, but the model shows promise and testing suggests that the theory behind the model is sound. As such, this project will be worth furthering through testing and refinement.**

I. Introduction

Complex systems are unavoidable in the realm of data science. Be it the stock market, the weather, or the physical condition of a human body, raw data taken from these systems can often be large, messy, and difficult to comprehend. However, some of these systems, like the human body, can be represented using a much simpler set of factors, such as the state of critical organ systems within the body.

A. Motivation

One such system that particularly interests us is human behaviour. Various efforts have been made to simplify and categorize human behavior into intuitive groups, but with varying degrees of success. One well-known example is the MBTI test [1], which has been criticized for its lack of scientific backing and questionable reliability. Despite this, some attempts to factor personality have proven to be more successful, such as the Big 5 personality test that aims to categorize personality into five distinct traits and has shown promising levels of validity. However, a common criticism for tests of this nature is their lack of reliability, especially given that the criteria and factors are often based on the creator's subjective intuition.

The objective of this research is to develop a customized model(outlined in the Methodology section) and assess its effectiveness in generating meaningful data representations. The performance of the model will be assessed based on its ability to learn a generalizable personality feature representation using an existing self-report survey dataset. Additionally, this study endeavors to create a personality factor representation that is entirely isolated from human bias.

B. Related Works

Autoencoders [2] are used to learn alternative or compressed representations of data. It does this using two neural networks, one that converts the data to a latent representation, and another that reconstructs the input data from the latent representation.

A common use for autoencoders is in data denoising. By training an autoencoder to represent the data in a limited latent space, noise that does not conform to the learned representation is discarded from the input data.

Another common use for autoencoders is in decomposing and representing complex physical systems in lower dimensions [3]. This utilizes an autoencoder to extract modes of nonlinear systems like fluid flow, allowing a linear transformation to be applied to the latent representation to reflect a similar transformation in the original system.

An example of this method uses an autoencoder to learn the sinusoidal properties of a turbulent flow system from a video recording of the system. A transformer can then be applied to the latent representation and reconstructed into a video frame representing the system after a fixed time interval.

C. Problem Definition

Autoencoders are generally bounded by a bias-variance trade-off. If an autoencoder's capacity to fit to the data is too high, the learned representation may not be meaningful and can instead overfit to the peculiarities of the dataset. This can make it difficult to analyze or transform the representation effectively. On the other hand, if the capacity of the autoencoder is too limited, the representation may not be accurately reconstructed to the original data.

In this study, a customized autoencoder-like model will be constructed in an attempt to overcome the flaws and trade-offs of traditional autoencoders. The aim of this model is to create generalized, meaningful representations without sacrificing the ability to accurately reconstruct data.

II. Methodology

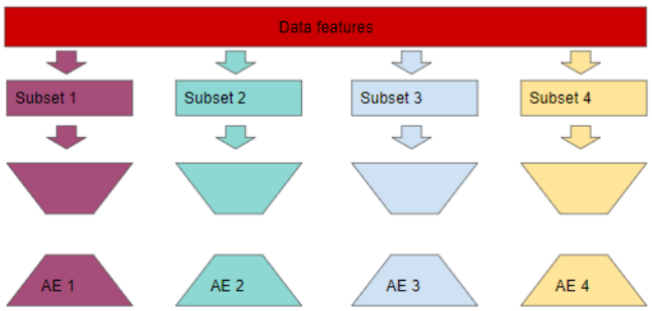

Fig. 1. Visual representation of how data is split and assigned to different encoders and decoders.

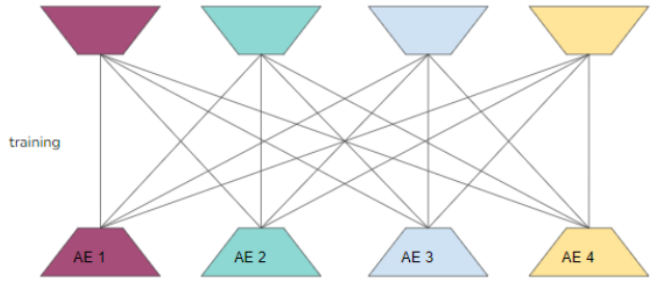

Fig. 2. Visual representation of encoders and decoders being cross trained.

1) Data:
The main dataset we will be using is the 16PF survey result dataset [4].
The data will first be cleaned by discarding unwanted features like country, name, gender, etc. entries that contain missing data or only contain one response value will be discarded.
Next, the data will be split into 3-5 subsets using randomized stratified sampling to ensure each subset contains a somewhat balanced set of questions based on the subgroups the survey questions were labeled with.
2) Proposed solution:
The custom model will consist of 3-5 encoder decoder pairs, each matched with a data subset, and all having the same latent space size. (see Figure 1-2)
When training, the model will iterate through all possible encoder decoder pairs for each train step, with the loss evaluating the mse between the decoder output and the data it is supposed to reconstruct. This is done to force all the encoders and decoders to learn the same latent space, as each encoder must represent the data in a way that can be decoded by all the decoders, and each decoder must learn to decode the latent representation of all the encoders. This also ensures that the encoders represent the data in a way that is generalizable and isn't easily affected by peculiarities that are specific to any one subset of data.
3) Evaluation:
The model will be evaluated on two metrics, which are applied to all encoder decoder combinations except the combination containing the encoder and decoder from the same set. This is done to focus the metrics on measuring how well the model generalizes the latent representation to non-identical data. The first metric is accuracy. This is measured by rounding and clipping the model's output to a valid survey response. It is then compared to an actual user's response to determine the accuracy of the model. The second metric is the model's binary loss, where the model will only be evaluated on how accurately it predicts the net sentiment(1-2 vs 4-5) of the survey response whenever the response isn't null(3).
4) Details/Analysis:
The optimal size of the latent space was determined by testing each latent space within a reasonable range(3-19) and recording the binary accuracy of the model trained with that latent space. It was determined that a latent space of 8 would be optimal as further increasing the latent space would result in a negligible gain in accuracy. (see Figure 3)
Using a fixed latent space, different model shapes and layer depths were tested. With a latent space of 8, it was found that the optimal shape was to have no hidden layers. However, hidden layers were used when using a latent space of 3.
The training progress with a latent space of 3 was also plotted to analyze how well each encoder's latent space was converging. The results of this analysis confirmed the theory that the different encoders would learn the same latent representation through the customized training function. (see Figure 4)

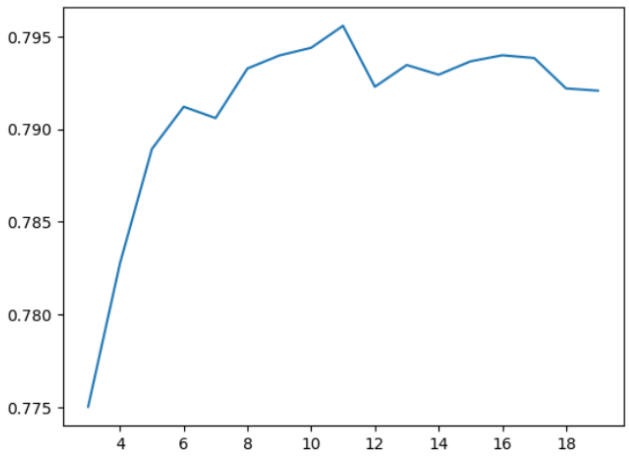

Fig. 3. binary accuracy of models trained with different latent space sizes.

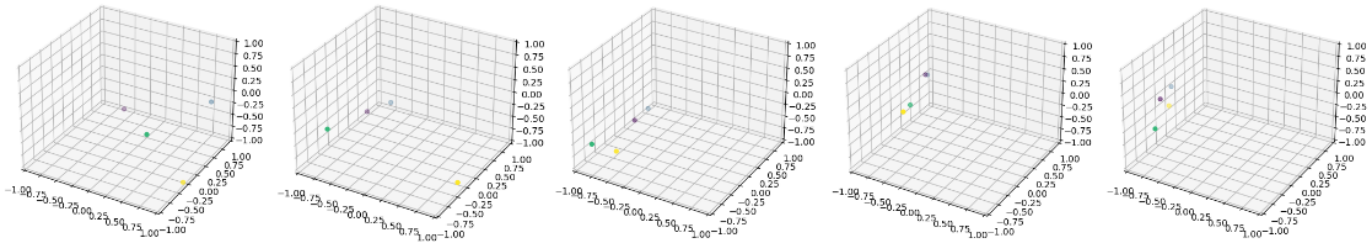

Fig. 4. A representations of four subsets in a data point converging as the model trains.

III. Results

TABLE I
MODEL RESULTS.

Latent space	Hidden layers	Accuracy	Binary Accuracy
8	0	40%	79%

As shown in table 1, the model appears to have performed poorly in terms of the accuracy metric, but decently well in terms of the binary accuracy. This shows us that the model could not predict the exact score a subject would respond to a survey question with, but could predict, generally, whether the subject agreed or disagreed with the survey question. While the model did not perform as well as expected, its binary accuracy shows promise, as it implies that the latent representation generated could represent the presence of personality features decently well, if not the exact degree to which they are present.

IV. Conclusion

Overall, this study has shown that the customized model is a viable method for creating alternate representations of data. It also shows that the theory behind the model is sound, as Figure 4 demonstrates that the latent spaces of the encoders and decoders converge to learn a shared representation of the data.

However, it has yet to be definitively shown that the customized model can create representations of data that surpass traditional autoencoders in terms of generalizability and meaningfulness. It has also yet to be shown how well the personality representation generated in this study compares to traditional personality tests like the Big 5.

In conclusion, this study has served as a good starting point for the development of the customized model. We have shown its viability, demonstrated the soundness of the theory behind it, and utilized it to create a representation of personality from a self report survey dataset to some degree of success. More testing and refinement will have to be done in order to polish and prove the customized model, and additional data processing and comparison will be required to fully develop an adequate personality representation.

References

[1] Stein, R, Swan, AB. Evaluating the validity of Myers-Briggs Type Indicator theory: A teaching tool and window into intuitive psychology. Soc Personal Psychol Compass. 2019; 13:e12434. https://doi.org/10.1111/spc3.12434

[2] Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. doi:10.48550/ARXIV.2003.05991

[3] Lusch, B., Kutz, J.N. & Brunton, S.L. Deep learning for universal linear embeddings of nonlinear dynamics. Nat Commun 9, 4950 (2018). https://doi.org/10.1038/s41467-018-07210-0

[4] Open psychology data: Raw Data from online personality tests. (n.d.). Retrieved March 13, 2023, from http://openpsychometrics.org/_rawdata/

Real-Time Sign Language Translation

Hendrix Gryspeerdt
Queen's University
hendrix.gryspeerdt@gmail.com

Liam Salass
Queen's University
liamsalass@yahoo.com

Xinran Wang
Queen's University
wxr.sammi@hotmail.com

***Abstract*—This report discusses the implementation of an American Sign Language (ASL) recognition model in Python, motivated by the need to break down communication barriers between the hearing-impaired and other communities. The problem is approached as an image recognition problem, and the team used various libraries such as TensorFlow, Keras, Mediapipe and sci-kit learn to build the model. Multiple different data sets were used. The initial models were convolutional neural networks (CNNs) and were later swapped out for detecting hand landmarks and training with support vector machines (SVMs) so that live predicting performance would increase. The significance of this model is highlighted by the lack of readily available software solutions for hand sign language recognition. The report concludes with the team's achievements in building an accurate ASL recognition model, which could potentially be further developed into a live-stream data solution with the use of more comprehensive data sets.**

I. Introduction

A. Motivation

There has always been a communication gap between the community of hearing- and speech-impaired individuals and the majority of people without those disabilities. American Sign Language(ASL), serving as the predominant sign language of deaf communities, is however challenging to learn and often unrecognized by the majority. To facilitate communication and promote the accessibility of sign language, we aim to develop a real-time sign language translation application that recognizes the ASL alphabet and translates them into readable text.

The motivation to solve this problem was initially established after reading the clients' proposal from Jomo Kenyatta University of Agriculture and Technology (JKUAT). Since there were no common-name, readily accessible software solutions providing sign language translation, their project aims to develop a system that enables real-time communication between hearing- and speech-impaired people and people who don't understand sign language. Pursuing a similar goal, we decided to build a real-time American Sign Language (ASL) recognition model in python that can be used in ASL learning scenarios.

B. Related Works

One commercial solution for ASL recognition the team came across was called SignAll. SignAll's products were not free to access, and the team should have contacted them to obtain a trial of their product. SignAll also mentioned on their website that they have an app called *Ace ASL* that is available on the apple store and google play. However, the app was nowhere to be found, leading the team to believe that this problem had not been solved yet, or at least the solution was unavailable as a commercial product.

In search for data sets of labeled ASL hand signs to train a model, the team came across multiple data sets, and one of those data sets was linked to an ASL recognition model hosted on AWS.

One data set we worked on (Kaggle data set #3) was used in a study by Moklesur Rahman et.al in 2019, which proposed a CNN model to recognize the numerals and alphabets in four publicly available ASL data sets. Their model took 64 x 64 pixels images into a CNN with 6 convolution layers and 3 pooling layers. It reached high accuracy of 99% on all the presented data sets, but the results didn't reach live-stream data. This has been the challenge for most ASL recognition models available, that they only achieve compromising results in their data sets, but perform unsatisfactorily for new data.

Other related works include iPhone app store ASL translators. These apps represent early attempts at leveraging technology to bridge communication barriers and highlight the significance of our model. The app signTranslator did single letter predictions, and the app Signer. Both applications performed abysmally, according to user reviews and our testing.

C. Problem Definition

Based on the poor performance of the readily available solutions (the apps listed above) we wanted to build our own, more effective solution to ASL alphabet recognition. Therefore, the goal of the project came was to recognize the American sign language letter from the alphabet being shown in a still image. Then, by extension of this capability, design control logic to allow the processing of live video feed into cohesive text.

The team initially identified the problem of ASL recognition and interpretation as an image recognition problem. The team has had past experiences using python along with various libraries such as TensorFlow, Keras, sci-kit learn, and others to build artificial intelligence models to solve similar problems.

II. Methodology

A. Design Process

1) begins with defining the "minimum viable product"
2) collect data for training models
3) do any necessary pre-processing of the data

4) code ai model and train with data, make sure that the model has a high accurracy on the testing and training data
5) test the model in real-time with a live video feed and see how well it works in practice
6) if it doesn't work in practice: modify live webcam pre-processing so that the image is better centered around the hand. If that still doesn't work go back to step 3. If you come back to this step a second time, go to step 2.
7) model works in practice. refine the control logic for determining when to add a letter prediction to the current string of character input and what the minimum confidence would have to be so that a prediction is not discarded.

B. Data set

different data sets we worked with and compare them in a table:

a) Sign Language MNIST: Kaggle data set MNIST The MNIST data set is composed of 27455 training and 7172 test data cases. Each training and test case represents a 28x28 pixel image (pixel1-784) with a label (0-25) corresponding to an alphabetic letter, excluding 9=J and 25=Z which required gesture motions. The data came from extending a small number of color images with various users and backgrounds doing hand signs. After the modifications, all the images were gray-scaled, cropped around the hand region of interest, and had low resolution. The data set was available on Kaggle and was stored as csv files.

b) Kaggle data set #2: Kaggle data set #2 The data set contains 27 classes: letters A-Z and unknown (images without hands in the frame). There was a total of 40500 images, 1500 for each class. It was not split into training and test data sets. The images were grey-scaled, cropped to the region of interest, and resized to the same dimensions, but were squished and tilted so that the hands in the frame were not of the same size. The images mostly had white sheet backgrounds, but part of it had messy backgrounds which resembled the real applications. The data set is the smallest among the three, but had the highest variability among the data, though it still lacks a presentation of unique hands. The data set was available on Kaggle and was stored as jpg files.

c) ASL Alphabet: Kaggle data set #3 The training data set contains 87,000 images in 200 x 200 pixels. It has 29 classes: 26 for letters A-Z (J and Z are presented without gestures) and 3 classes for SPACE, DELETE, and NOTHING. The NOTHIHG class represents a blank wall, which was equivalent to Mediapipe detecting no hand in the frame. These 3 classes allow transitions between spelling different letters in real-time applications and classifications. The test data set only contains 29 images, one for each class, which allows us to test the model against our live-stream data. The data in the data set are color images cropped around the hand region of interest. However, it is worth noting that the images all represent similar hands with similar backgrounds, providing little variety between the data. There is also an apparent lack of rotated hand signs. This resulted in letters such as 'L', 'D', and 'X' not being classifiable from their side views. The data set was available on Kaggle and GitHub and was stored as jpg files.

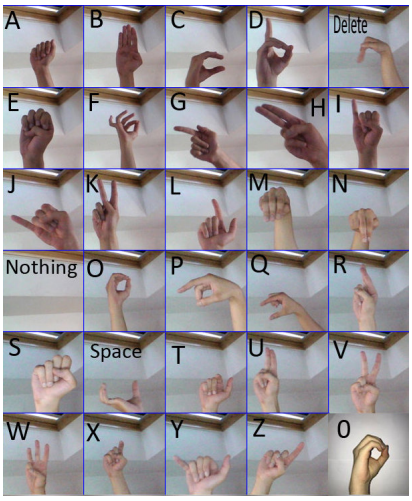

Fig. 1. ASL alphabet data set.

C. CNN failed to predict live-stream data

potential reasons using CNN on images didn't work:

- CNNs are better designed for recognizing 2D features and hand gestures are 3D features. It works for detecting hands in a frame but may fail to extract the unique 3d shape of different hand signs in a live-stream feed.
- Bigger data sets with more variety are needed: the data in all three training data sets lacked variety. The hands were similar or had similar backgrounds, so the model fails when the live-stream input data are vastly different.
- When dealing with real-time data the number of possible hand orientations representing the same letter differ largely, therefore to extract highly specific hand orientation in 3D space did not lend itself to be accomplished by a convolutional neural network.

D. Using Mediapipe to detect hand-landmarks

Mediapipe is a hand-detection open-source library produced by Google. It was utilized to convert a data set containing hand images into a new data set containing hand landmarks positions in 3 dimensions. There are 21 hand landmarks that media pipe would generate on a photo, however, only 20 points were used as inputs to the SVM. This was done by subtracting the position of the wrist landmark from all 20 other landmarks to get their relative positions to the wrist, as opposed to their relative position in the frame. The newly generated data set was then used as the training data for the SVM model. The library was also used to detect where the hand is in the frame and crop the hand's region to be passed to models (CNN models) as square images. That way, all images passed to the model avoid being compressed, and fit the models' input shape (in the case of the CNN). For the aforementioned reasons,

Mediapipe was a necessary component to the success of our project.

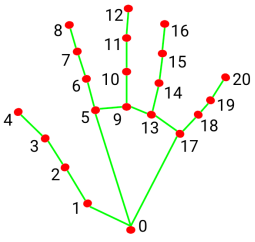

Fig. 2. 21 Mediapipe hand landmarks.

E. Support vector machines

Support vector machines are effective at classifying sets of high-dimensional vectors. The intuition behind using a support vector machine to classify hand sign gestures is as follows. In the case of the hand landmarks generated by Mediapipe (see image above), the set of 3D points that form the hand orientation can be flattened into a vector representing a position in a high-dimensional space. For example, the space of hand landmark orientations that would be considered to be an A would form some region in the high dimensional space, and support vector machines are inherently effective at classifying these types of regions. This model corresponds with our common way of understanding hand signs - according to the relative locations of the fingers, instead of the exact image of a hand. It is better to use one model to locate general hand features (Mediapipe), then another to parse those features into different hand signs (SVM).

III. Results

All the code and demos can be found on our teams github page.

TABLE I
TABLE SHOWING THE EXPERIMENTAL RESULTS FROM TRAINING THE ASL RECOGNITION MODELS ON THE LISTED DATA SETS.

Model	Model Type	Data Set	Accuracy
ASL_model1	CNN	MNIST from Kaggle	91.37%
ASL_model2	CNN	MNIST from Kaggle	99.74%
ASL_model3	CNN	Kaggle data set #2	99.84%
svm_landmark_model1	SVM	Kaggle data set #3	98.61%
svm_landmark_model2	SVM	Kaggle data set #3	98.58%

Even though all the models were able to perform well on their own data sets, they did not all perform equally well in practice. When running live video, all of the convolutional models had issues with being overconfident at times when there was a hand in the frame, but the hand was giving hand sign that was not recognized by ASL. However, the support vector machine models, which ran off of the hand landmark data, were as accurate as during training as they were during real-time testing. The SVM models were also much less computationally slow to run, allowing for a smoother frame rate during a demo.

IV. Conclusion

In conclusion, our ASL prediction model has demonstrated exceptional performance in live predictions compared to other related works, indicating its significance in the field of American Sign Language recognition. Our model's performance is attributed to using an SVM-based classification using extracted hand landmarks instead of a CNN-based approach. This approach allowed us to overcome some of the limitations of the datasets available and allowed for fewer parameter inputs to the model. Resulting in a more efficient and accurate model. Our model's accuracy in predicting hand gestures in real-time is a testament to the effectiveness of our approach.

Our immediate next steps for improving our ASL prediction model involve increasing prediction accuracy and improving the control logic for creating strings of letters to make words. To achieve this, We will create our own dataset to ensure that it is more accurate and complete. The data sets our group could find all had inherent faults, ranging from too small data sets, and data sets with low variance in images, to having incorrect labeling for whole single letters. By including more images of hand signs taken from vertically rotated angles in the data set, the model will better predict letters that have been rotated.

Additionally, we would like to expand the current model's alphabet to include the digits 0-9. A more comprehensive data set would be required including signs for the digits. We will then integrate this data into our model and train it to recognize these new features accurately. This model would still be using a rolling average of the previous frames's predictions to predict the current letter.

Looking further ahead, we would like to develop an app that can perform these predictions, making ASL communication more accessible and efficient. With these improvements, we are confident that our ASL prediction model will become a valuable tool for promoting the learning of ASL and increasing the ability to communicate in sign language with people with hearing impairments.

References

[1] [Name]. [Name of item]. [location], [year]
[2] Sangeetha Rao. Signer - AI Sign Translator. Apple App Store, 2022.
[3] Andrew Ryan. signTranslator. Apple App Store, 2022.
[4] Google. Mediapipe. Github, 2022
[5] Google. Mediapipe - Hands. Mediapipe website, 2022
[6] Md. Moklesur Rahman, Md. Shafiqul Islam†, Md. Hafizur Rahman, Roberto Sassi§, Massimo W. Rivolta, Md Aktaruzzaman. A New Benchmark on American Sign LanguageRecognition using Convolutional Neural Network. ASL Essay, 2019.
[7] Kaggle username: tecperson. Sign Language MNIST. Kaggle Website, 2018.
[8] Muhammad Ali. Sign Language for Alphabets. Kaggle Website, 2020.
[9] Akash. ASL Alphabet. Kaggle Website, 2018.
[10] Quantiphi. Sign Language Recognition. AWS Marketplace website, Na.
[11] Zsolt Robotka, János Rovnyai, János Rovnyai, Sean, Jesada Pua, Judit Tóth-Molnár, Dávid Retek, Derek Frank, Mihály Pintér, Dávid Pálházi, Shreya Bhattacharya, Helga Mária Szabó, Éva Sáfár. SIGNALL, Signall Main page, 2022

Towards the Responsible Development of AI

Seth Grief-Albert
Queen's University
seth.griefalbert@queensu.ca

Cyrus Fung
Queen's University
19thcf@queensu.ca

Aamiya Sidhu
Queen's University
21as226@queensu.ca

I. Introduction

Progress in Artificial Intelligence (AI) is accelerating at an unprecedented rate. The introduction of AI applications into the public domain will have profound effects on several institutions across healthcare, industry, and beyond. This necessitates reevaluating the ways in which revolutionary technology has been and continues to be developed. Social media offers a cautionary tale, that entrenched attitudes towards innovation at any cost have led to negative consequences. This attitude is incompatible with developing technology ethically. A shift in focus from rapid development to responsible development is necessary, calling for a more thoughtful approach to the creation and deployment of AI.

II. Radical Disruption

We are at a point where moving fast and breaking things has solidified itself as a driving force behind innovation. The biggest disruptors in AI are the same platforms that used this very idea to emerge as global conglomerates.

At the outset of the social media revolution, the mantra "move fast and break things" voiced by Mark Zuckerberg reflected the budding entrepreneurial attitude towards innovation [1]. This idea emerged as social media was being built as a call for fast progress and repeated iteration, at the cost of careful deployment. Social media represented a paradigm shift in communication towards a personalized network with unprecedented monetization potential. Data fuels social media. In the process of engaging with platforms, users generate information that is applicable to advertising. In 2020, over 97% of Meta's revenue was from advertising [2]. With a major market share and a highly secure profit stream, it is entirely rational for businesses like Meta to keep advertisers as clients, and make decisions with the interests of advertisers in mind.

At the same time, when technological progress moves faster than regulation and ethical guidelines, massive systems with murky purposes and emergent phenomena can arise. Without guardrails, negative consequences can easily manifest. A prominent phenomenon on social networks is the echo chamber. Echo chambers are characterized as radically exclusive social structures, with the potential to harm vulnerable online communities [3]. With little oversight and algorithms that prioritize personalization, echo chambers have established themselves throughout social media [4]. Algorithmic personalization is based on the interactions and engagement of a user, and content that someone is more likely to engage with is fed to their home screen [5]. Thus, pre-existing biases can take root, presenting a warped view of the world. During the COVID-19 pandemic, health misinformation proliferated online throughout social media [6]. Additionally, trust in social media has risen to a point where many people solely engage with it for their news [7]. The disruption of powerful technology over time paints a picture that moving fast and dismissing consequences is misaligned with the goals of beneficial and ethical application.

III. The State of AI

A. Bias in AI

From 2015 to 2021, the compound growth rate in AI-related patents was 76.9% [8]. Implementing novel developments in AI as fast as possible has produced numerous gaps, because the positive performance of AI is predicated on good data. Faulty data ravages applications in criminal law, advertising, recruiting, and computer vision [9]. It is thus necessary to evaluate the approach taken by enterprises towards data.

A black box system is characterized by inaccessible operations and explanations of outputs. In machine learning, a subset of AI, systems are created purely from data for use in algorithmic processing [10]. The output of complex systems in AI can be dictated by billions of components, potentially transcending human understanding of internal operations. Yet, the societal impact of these systems is significant. One such case is the COMPAS algorithm, which has been used to determine the likelihood of a criminal becoming a repeat offender [11]. These systems rely on historical data from eras with different laws and political views, making them highly susceptible to human bias [11]. When flawed data is applied to black box systems, a lack of transparency for developers and consumers has negative impacts.

Algorithmic bias in AI is defined as "systematic and repeatable errors in a computer system that create unfair outcomes, such as privileging one arbitrary group of users over another" [12]. This bias has become one of the most pressing issues in modern AI applications. Bias is especially relevant in healthcare, where impacts can greatly affect the lives of individuals. Overarching problems with the use of AI in healthcare involve fairness, lack of context, and non-explainability. As it stands, inequities within western society and healthcare systems result in a lack of quantitative metrics of fairness. This leaves interpreters of medical evaluations responsible for reducing bias [13]. Healthcare and treatments for individuals are specific to their context which includes an individual's environment, culture, socioeconomic status, lifestyle choices and genetics

[13]. Given the number of variables, intersectional impacts, and lack of data for major population groups, AI applications in healthcare currently lack the ability to provide accurate data for underrepresented populations [13]. Because of the power and influence of AI algorithms, black box systems can leave data scientists, clinicians, and patients without knowledge of how predictions are being made. This has detrimental effects as progress accelerates and AI becomes increasingly present in healthcare [13]. The potential consequences of inequitable training data are severe. Today, several facial recognition training datasets mostly contain images of white males [9]. In tests with recent image recognition models such as CLIP, Black people could be misclassified as non-human [8].

B. Intellectual Property

The massive scale and black-box nature of generative models further complicates legal and ethical concerns regarding intellectual property. Models such as GPT-3, Copilot, and DALL-E 2 are trained on large amounts of data scraped from the internet, which inevitably contains copyrighted data [19]. A user could plausibly generate an output that closely resembles copyrighted data, by fine-tuning GPT-3 (a language generation model) with samples from a specific author in order to pass off its output as an original product. Models can also infringe on copyright by directly regurgitating training data in their outputs, potentially misleading users into thinking that it is a novel creation and not stolen intellectual property. For example, Copilot has been found to generate large sections of copyrighted code verbatim, leading to a class-action lawsuit against its creators for violating the rights of the millions of GitHub users who published code under an open-source license [20].

As awareness of the risk that generative models pose to intellectual property rights grows by the day, most companies filter prompts to dissuade claims that they are facilitating theft or turning a blind eye. Unfortunately, user-level interactions only represent the tip of the iceberg of copyright infringement in AI. A lack of regulation and oversight gives companies ample opportunities to exploit loopholes that allow them to violate copyright to generate profit, while also shielding themselves from accountability. One such loophole is "data laundering," where for-profit companies strengthen their claims of fair use and profit off of copyrighted information by outsourcing the training and dataset collection process to non-profit entities [12]. Consider the text-to-image latent diffusion model Stable Diffusion by Stability AI, which has already been implicated in several copyright infringement lawsuits and is widely controversial amongst human artists [15]. One might believe that Stability AI was the creator of their flagship model, but a quick look at its GitHub repository reveals that it was trained by researchers at the Ludwig-Maximilians University of Munich using "a generous compute donation from Stability AI" [16]. The dataset used to train the model was not collected directly by Stability AI either: instead, it came from a non-profit organization called LAION, which also received compute resources from Stability AI [17]. The degree of separation that data laundering creates protects companies in two ways. Firstly, since these models were technically created for non-profit or academic purposes, most courts would consider them as falling under fair use. Secondly, in the event of unwanted scrutiny, public attention or legal liability can be redirected from the parent corporation to the non-profit organization that created the model with its funding and guidance. This allows corporations to convert what is ostensibly academic research into a monetized product while minimizing legal risk, as seen with Stability AI's product DreamStudio (essentially a consumer-friendly wrapper around the Stable Diffusion API). Without the appropriate regulatory oversight in place, continuing to move fast and break things in AI will only lead to unprecedented levels of intellectual property theft, harming both consumers and producers who cannot afford to keep up.

IV. Responsibly Developing AI

AI can only move as fast as the data that powers it and the regulation that guides it. Key to the responsible development of AI is transparency and accountability to stakeholders. The nature of black-box technology in machine learning applications poses a barrier to transparency and explainability. Transparency in data sourcing is an important aspect of ethical AI deployment. Data is the backbone of AI. This leads to external failure when data is non-representative, perpetuating human bias and leading to real harm [9]. Creating representative, equitable data is difficult when many companies outsource this task to third parties, thus obscuring responsibility.

Accountability in AI must be applied to every step of system development, from initial design to system monitoring. An accountable AI system encompasses four dimensions: strong governance, understanding the data, clear performance goals, and continuous monitoring [18]. The governance of a system includes ensuring the work-force is well-rounded with diverse perspective, has broad stakeholders, and strong risk-management. Accountability also requires documentation at every level. This includes technical specifications, system compliance and output, potential issues, and performance assessments [18]. Documentation must be available to stakeholders along with design and operation information. Most significantly, strong governance puts emphasis on the responsibilities of the authorities that control the deployment of the system. Accountability in a system entails a thorough understanding of the data that is used to create a model, and that understanding remains while the system is in operation. In addition, reliability and representativeness of the data must be understood to look for bias, inequities, and societal concerns from applications of the system [18]. Performance goals must be clear, well-documented, and stable from the first stages of development until the system is operational [18]. This includes performance assessments of the overall system and its individual components. The monitoring of a system entails having a set range that allows the system to "drift" and must allow for continuous questioning about the function and importance of the system [18]. Long-term monitoring

assessment and changes to a system must be done to assess if the system is still working and more importantly, if the system is still required.

V. Conclusion

Charging into the future as fast as possible leaves little time for true understanding of the societal implications of AI. Although a significant body of work has already been advanced regarding AI safety, concrete implementations by governments and corporations are still lacking. This may not just be a result of inertia or competing incentives, but could also be a natural product of the black-box nature of many AI systems. This makes defining key factors such as transparency, accountability, and reliability a difficult and possibly even intractable task. Without a clear and unambiguous means of evaluating outcomes of AI applications for both government and corporate bodies, the window of responsible development of AI appears to be shrinking rapidly. Legally enforced regulatory frameworks and bodies may be necessary to ensure that AI systems adhere to the aforementioned metrics. However, these frameworks must be highly flexible to handle ambiguity, and also to keep up with the rapid evolution of AI safety going forwards. By shifting the focus from innovation at any cost to an attitude of discretion and foresight, the revolutionary potential of AI can be harnessed for the benefit of all.

References

[1] J. Liles, "Did Mark Zuckerberg say, 'move fast and break things'?," Snopes, 29-Jul-2022. [Online]. Available: https://www.snopes.com/fact-check/move-fast-break-things-facebook-motto/. [Accessed: 12-Mar-2023].

[2] S. Dixon, "Annual advertising revenue of Meta Platforms worldwide from 2009 to 2022," Statista, 13-Feb-2023. [Online]. Available: https://www.statista.com/statistics/271258/facebooks-advertising-revenue-worldwide/. [Accessed: 03-Mar-2023].

[3] C. T. Nguyen, "ECHO CHAMBERS AND EPISTEMIC BUBBLES," Episteme, vol. 17, no. 2, pp. 141–161, 2020.

[4] C. T. Nguyen (2021). How Twitter gamifies communication. In Jennifer Lackey (ed.), Applied Epistemology. Oxford University Press. pp. 410-436.

[5] Meta, "What kinds of posts will I see in Feed on Facebook?," Facebook Help Center. [Online]. Available: https://www.facebook.com/help/166738576721085. [Accessed: 03-Mar-2023].

[6] V. Suarez-Lledo and J. Alvarez-Galvez, "Prevalence of Health Misinformation on Social Media: Systematic Review," Journal of Medical Internet Research, vol. 23, no. 1, Jan. 2021.

[7] J. Liedke and J. Gottfried, "U.S. adults under 30 now trust information from social media almost as much as from national news outlets," Pew Research Center, 27-Oct-2022. [Online]. Available: https://www.pewresearch.org/fact-tank/2022/10/27/u-s-adults-under-30-now-trust-information-from-social-media-almost-as-much-as-from-national-news-outlets/. [Accessed: 03-Mar-2023].

[8] D. Zhang et al., "The AI Index 2022 Annual Report," AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University, March 2022.

[9] N. Turner Lee, P. Resnick, and G. Barton, "Algorithmic bias detection and mitigation: Best practices and policies to reduce consumer harms," Brookings, 22-May-2019. [Online]. Available: https://www.brookings.edu/research/algorithmic-bias-detection-and-mitigation-best-practices-and-policies-to-reduce-consumer-harms/. [Accessed: 03-Mar-2023].

[10] C. Rudin and J. Radin, "Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition," Harvard Data Science Review, 22-Nov-2019. [Online]. Available: https://hdsr.mitpress.mit.edu/pub/f9kuryi8/release/8. [Accessed: 03-Mar-2023].

[11] T. Cassauwers, "Horizon Magazine," Horizon, 01-Dec-2020. [Online]. Available: https://ec.europa.eu/research-and-innovation/en/horizon-magazine. [Accessed: 03-Mar-2023].

[12] "Research guides: Algorithm bias: Home," Home - Algorithm Bias - Research Guides at The Florida State University, 23-Sep-2021. [Online]. Available: https://guides.lib.fsu.edu/algorithm. [Accessed: 03-Mar-2023].

[13] T. Panch, H. Mattie, and R. Atun, "Artificial Intelligence and algorithmic bias: Implications for health systems," Journal of global health, Dec-2019. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6875681/. [Accessed: 03-Mar-2023].

[14] A. Baio, "AI data laundering: How academic and nonprofit researchers shield tech companies from Accountability," Waxy.org, 30-Sep-2022. [Online]. Available: https://waxy.org/2022/09/ai-data-laundering-how-academic-and-nonprofit-researchers-shield-tech-companies-from-accountability/. [Accessed: 01-Mar-2023].

[15] A. Martin, "Lawsuits over stability AI's stable diffusion could threaten the future of AI-generated art," Business Insider. [Online]. Available: https://www.businessinsider.com/stable-diffusion-lawsuit-getty-images-stablility-ai-art-future-2023-1. [Accessed: 28-Feb-2023].

[16] Stability AI, "Stability-ai/stablediffusion: High-resolution image synthesis with Latent Diffusion Models," GitHub. [Online]. Available: https://github.com/Stability-AI/stablediffusion. [Accessed: 01-Mar-2023].

[17] R. Beaumont, "5B: A new era of open large-scale multi-modal datasets," LAION. [Online]. Available: https://laion.ai/blog/laion-5b/. [Accessed: 01-Mar-2023].

[18] S. Sanford, "How to build accountability into your ai," Harvard Business Review, 30-Aug-2021. [Online]. Available: https://hbr.org/2021/08/how-to-build-accountability-into-your-ai. [Accessed: 03-Mar-2023].

[19] C. Xiang, "Ai is probably using your images and it's not easy to opt out," VICE, 26-Sep-2022. [Online]. Available: https://www.vice.com/en/article/3ad58k/ai-is-probably-using-your-images-and-its-not-easy-to-opt-out. [Accessed: 03-Mar-2023].

[20] R. Losio, "First Open Source copyright lawsuit challenges github copilot," InfoQ, 18-Nov-2022. [Online]. Available: https://www.infoq.com/news/2022/11/lawsuit-github-copilot/. [Accessed: 03-Mar-2023].

Utilizing Sentence Transformers To Preform Semantic Searches

Bryson Reid
Queens University
19BWR@queensu.ca

Henry Xiu
Queens University
henryxiu2005@gmail.com

Joseph Moraru
Queens University
jos.moraru@gmail.com

Eric Lange University
Queens
17ewl@queensu.ca

***Abstract*—A Bert based sentence transformer model was trained and applied to preform semantic searches in an advanced word find web application to increase learning productivity when searching through PDF documents.**

I. Introduction

Sentence Transformers are a new type of NLP tool which can tokenize entire sentences for many different use cases. One such use case is sentence semantic matching, this means that a machine learning model can deduce how similar two phrases are in meaning regardless of sentence length and specific term matching. This report highlights the general process to training a sentence transformer model and specifically how a model was trained and used to preform semantic searches in a PDF document.

A. Motivation

The general word find that is found in most products simply consists of looking for the exact phrase input by the user, while this can be very helpful if you want to search for something you know for sure is in the text it is not very useful when all you have is a general idea of what you're looking for. When learning new skills and researching topics there are many situations where you might have a general topic to search for and a classical word find may return too many terms or none at all. An advanced word find A.K.A semantic word search returns similar phrases in a document to the key term, this allows for more complex and detailed searches in a document that may not actually exist. To achieve this, a sentence transformer model can be trained and implemented to identify similarity between sentences.

B. Related Works

The sentence transformer model is a relatively recent addition to nlp. It utilizes the standard transformer model along with some extra layers to put sentences into a vector space for comparison. Project we have referenced used the pre-trained BERT transformer model, an averaging layer, and a condensing layer to form the model. Many open source projects focus either on the training stages of the model or the operation and don't include the full process which is what this project aimed to do. Hugging Face gives the example of using a sentence transformer to create a search engine for frequently asked questions in GitHub and TensorFlow demonstrates a project doing similar work but for news headlines [1] [2]. This Project aimed to create a functioning web app that utilizes a trained sentence transformer model to perform semantic searches on uploaded PDF's. This project Works very similarly to previous projects published by Hugging Face and tensor flow through applying the model to sentences in a PDF and searching through this list of sentences.

Hugging face and the Sentence-Transformers library documentation provide excellent instruction on the ways to train this type of model. Firstly the model itself can consist of multiple modules, that is, layers which execute consecutively. These layers include a word embedding model and a pooling model which can also include dense, bag of words, and convolutional layers. The model can then be trained on different formats of data sets listed below [3].

1) Example is a pair of sentences and a label indicating similarity. The label can be either an integer or float.
2) Example is a pair of similar sentences without a label.
3) Example is a sentence with an integer label. This data format is easily converted by loss functions into three sentences (triplets) where the first is an "anchor", the second a "positive" of the same class as the anchor, and the third a "negative" of a different class. Each sentence has an integer label indicating the class to which it belongs.
4) The example is a triplet (anchor, positive, negative) without classes or labels for the sentences.

A loss function is then required to recognise the similarity between sentence embedding. Different loss functions are required for the different formats of data sets and some loss functions may be more suited to the specific task required of the model which is why one form of data may be preferable to another [3] [4].

Once all of these factors have been decided then the code is quite simple to finally create and train the sentence transformer model.

C. Problem Definition

The problem our team tried to solve was to train and implement an effective sentence transformer model that can be used to preform semantic searches on a PDF. This tool should be easily accessible, simple to use, and accurately return semantic terms. This tool is meant to be used by students who find themselves looking through large technical and none technical documents.

Some problems that were faced in this project included training a sentence transformer model, effectively delimiting a PDF into its sentences in Python, implementing the model in Python, and creating a front end interface that can utilize the Python back end.

II. Methodology

There were two main sections that went into this project, training the sentence transformer, and implementing it. Based off the previous research, the specific training methods are detailed below. `Training`

1) First, the model needed to be formed using three layers, a BERT transformer model, an average pooling layer, and a condensing layer.
2) Second, the snli data set was imported for the model to train on.
3) Third, the training data was extracted from the data set and formatted properly for the model to train.
4) Finally, the model was trained on over 500k training data points which consisted of two sentences and a 0,1 or 2 indicating if they contained the same, similar, or opposite meaning.

`implimentation`

1) First, PDF's were imported and delimited using the PDF miner six library.
2) Second, the sentence transformer model was imported.
3) Third, functions were defined to return semantic search results from the model.
4) Lastly, a front end web application was created so users could upload a PDF, and utilize the functions which were connected to the front end.

III. Conclusion

This report demonstrates an effective use case for sentence transformer models and how to train one from scratch. Implementing a sentence transformer to preform semantic searches on a document allows for increased productivity when searching for information in a large document where classical key term searching fails.

References

[1] Hugging Face, "Semantic search with FAISS," The Hugging Face Course, 2022. https://huggingface.co/course/chapter5/6?fw=tf#semantic-search-with-faiss

[2] TensorFlow, "Semantic Search with Approximate Nearest Neighbors and Text Embeddings," Jan. 13, 2021. 'https://www.tensorflow.org/hub/tutorials/semantic_approximate_nearest_neighbors'

[3] O. Espejel, "Train and Fine-Tune Sentence Transformers Models," Hugging Face, Aug. 10, 2022. https://huggingface.co/blog/how-to-train-sentence-transformers#: :text=To

[4] N. Reimers, "Training Overview," Sentence-Transformers, 2022. https://www.sbert.net/docs/training/overview.html