PROCEEDINGS

of the Canadian Undergraduate Conference on Artificial Intelligence

MARCH 6-7, 2021 | ONLINE

Artificial Intelligence.
Real Change.

cuc ai 2021

Copyright

Design by Katie Lu.

Best Paper Awards

Of all the submissions, a shortlist of the top 10 papers were selected. These 10 papers were ranked by our award selection team, comprised of professors in the field of AI, on the basis of quality of work, novelty of the work, and quality of the paper.

Thank-you to Christian Muise, Farhana Zulkernine, Felicia Magpantay, and Tamas Ungi for your time and expertise in ranking the best papers.

Congratulations to the authors of the following top 5 best papers:

1 Android Waste Classification

2 Predictive Diabetic Risk Modeling

3 Autonomous Highway Simulation with Smart Cruise Control

4 Virtual Assistant Attention Detection

5 Music Genre Classification Using K-Nearest Neighbours And Neural Networks

Table of Contents

Sponsor Acknowledgement

We'd like to take a brief moment to thank this year's conference sponsors, without whom CUCAI 2021 would not be possible. Thanks to your generosity, we were able to host the largest, and most successful CUCAI to date, and the speakers, panelists, workshop leaders and industry booth representatives that you provided us with were all instrumental to the outstanding quality of the conference this year. You all helped deliver an unforgettable experience to hundreds of passionate delegates from across Canada, and you've helped us get one step closer to our goal of fostering a unified, Canadian undergraduate AI landscape through our conference.

We'd like to give a special thank you to our Title Sponsor: The Queen's University Faculty of Engineering and Applied Science (FEAS). We are forever grateful for your support. Second, we'd like to thank our Gold Sponsors: Northeastern University Toronto, and TELUS. Lastly, thank you to our Silver Sponsors: bld.ai, WeTraq, MOSEA, Vector Institute, Kinaxis, Distributed Compute Labs (DCL), Google Cloud and The Dunin-Deshpande Queen's Innovation Centre (DDQIC).

Thank you once again for all your support, and we hope you'll be able to join us again for CUCAI 2022!

TITLE SPONSOR

Queen's University | Faculty of Engineering and Applied Science

GOLD

Northeastern University Toronto

TELUS

SILVER

wetraQ

Distributed Compute Labs

MOSEA

Google Cloud

Vector Institute

bld.ai

Kinaxis

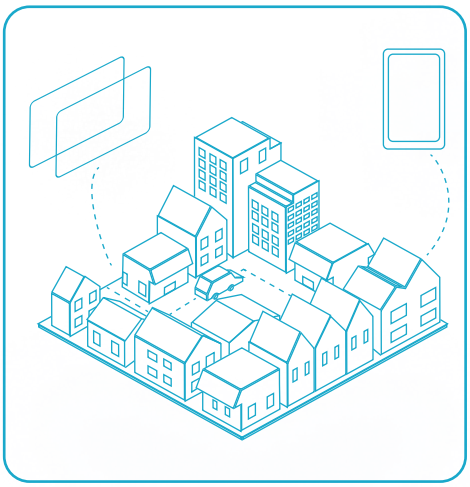
Dunin-Deshpande Queen's Innovation Centre

Conference Organization

CO-CHAIRS

Berkeley Wilson, Queen's University, Canada

Max Bennett, Queen's University, Canada



MARKETING

Katie Lu, Queen's University, Canada

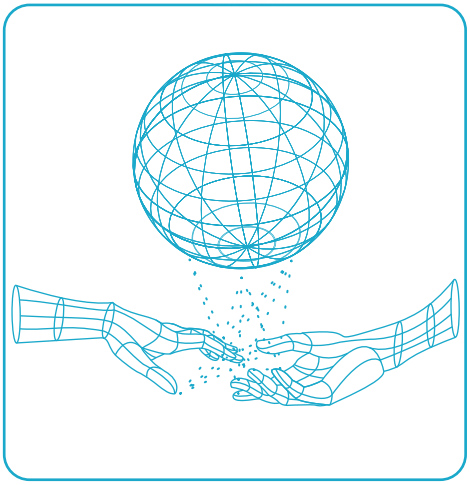
Parker Rowe, Queen's University, Canada

LOGISTICS

Catherine Wu, Queen's University, Canada

Jack Grebenc, Queen's University, Canada

Will Stewart, Queen's University, Canada



DELEGATES

Ellie Mehltretter, Queen's University, Canada

Jack Perry, Queen's University, Canada

PARTNERSHIPS

Elan Bibas, Queen's University, Canada

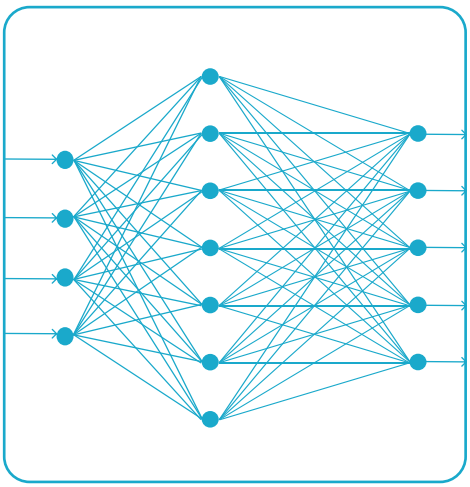
Aman Dhaliwal, Queen's University, Canada

Anusha Mane, Queen's University, Canada

Brendan Nugara, Queen's University, Canada

Esteban Perez, Queen's University, Canada

Raisa Sayed, Queen's University, Canada



CHIEF EDITOR

Jacob Laframboise, Queen's University, Canada

Foreword from CUCAI

On March 6th and 7th, 2021, the third annual Canadian Undergraduate Conference on Artificial Intelligence (CUCAI) was hosted in an online format using Hopin, to make CUCAI 2021 the largest and most accessible to date, while protecting the health and safety of our attendees. After successful in-person conferences in both 2019 and 2020, this year's event featured 500+ delegates and industry representatives from across Canada and beyond. Highlights from the weekend included presentations from renowned industry leaders on a variety of topics related to our conference theme, "AI for Good," engaging workshop events, our Women in AI Leadership and Ethics in AI panels, the Student Design Showcase and Industry Showcase events, and the CUCAI Pitch Competition. Students had the opportunity to showcase their hard work throughout the past year on their design projects, and learn about the opportunities available for them to get involved within the AI industry, while industry members discussed the work that their respective companies are engaged in to make the world a better place through the use of artificial intelligence, and had the opportunity to learn about the significant growth of the undergraduate AI landscape in Canada in recent years.

Throughout the course of the weekend, attendees had the chance to see speaker presentations from Ron Bodkin, the VP of AI Engineering and CIO at the Vector Institute, Inmar Givoni, the Former Director of Engineering at Uber ATG, Laurence Moroney, the Lead AI Advocate at Google, David Hayes, the Founder and CEO of the Autonomous Vehicle Organization (AVO), and Shivon Zilis, the Project Director at Neuralink's office of the CEO. Attendees also heard from Caroline Lair, the Founder of The Good AI and Co-Founder of Women in AI, Shirin Farrahi, a Principal Software Engineer at Cadence Design Systems, and Ania Halliop, a foamier Senior Engineering Manager at Uber ATG, on the Women in AI Leadership Panel, and Valerie Morignat, the Founder and CEO of Intelligence Story LLC, Rishi Behari, a leadership Instructor at the Smith School of Business, MMAI program, and Ron Bodkin on the Ethics in AI Panel. Finally, through the Industry Showcase, and Speed Networking section of the event, students had the ability to network with esteemed industry representatives from companies working in the AI/ML field. CUCAI 2021 was made possible by the generous donations from this year's sponsors, including our Title sponsor, the Queen's Faculty of Applied Science. The full list of sponsors can be found on the Sponsors Acknowledgement page.

The Student Design Showcase event took place for the third time this year, and it allowed design team members from QMIND and Western AI to present on the accomplishments of their year-long AI design projects to other conference delegates and industry professionals. The outstanding calibre of these projects is demonstrated by the quality of the articles found in these proceedings. CUCAI strongly encourages other AI student groups in Canada and beyond to reach out and participate in the Student Design Showcase event in future years.

Android Waste Classification

Sean Sutherland[1], Richard Robinson[2], Emily Mendelson[3], Tung Pham[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: sean.sutherland@queensu.ca
2 e-mail: 16rer@queensu.ca
3 e-mail: 17ecm1@queensu.ca
4 e-mail: 18tp12@queensu.ca

Abstract: *The gradual growth of solid waste in the urban area has been and is becoming a great concern for human health, and could result in environmental pollution and may be hazardous to humanity if not properly disposed. With that being said, an advanced waste management system is necessary to manage a variety of waste materials. The most important step of waste management is the separation of the waste into their designated categories. This process is usually done by manually hand-picking and sorting them into their designated bins. In order to save time and simplify the process, Queen's Waste Wizard introduces a waste classification computer vision model, which is developed using a pre-trained residual net (EfficientNet-B3) Convolutional Neural Network model, a machine learning tool. The model is used to classify waste into different groups such as* ***Blue Recycling****,* ***Grey Recycling****,* ***Landfill*** *and* ***Organics****. Our first proposed system was able to achieve an accuracy of 98% on the validation dataset. An overall 80% accuracy was achieved on our testing dataset with more realistic examples with background noise. This method is expected to be faster, by implementing the proposed system without or little human involvement.*

1. INTRODUCTION

1.1 Motivation

Globally, annual solid waste is expected to reach 2.2 billion tons by 2025 [1]. Improper waste management may lead to huge economic, environmental, and public health issues. As a result, there is a clear need for proper waste management within public spaces where a large proportion of waste is improperly discarded.

In 2016, the waste diversion rate of Queen's University was measured to be 43.23%, however, over 85% of the current waste stream was composed of items that can be diverted from landfills [2]. While Queen's University has already deployed a waste lookup application, this tool requires a high amount of user input. In contrast, computer vision models only require the user to take a single image of the waste item for classification. This type of application may help increase the waste diversion rate on campus by reducing the amount of misclassified waste.

While there are existing computer vision models for waste classification, there is limited use of such models within public spaces. Many models including those using AlexNet [3] and Inception-ResNet [4] have been trained for waste classification, however, these models are large and require a large number of parameters to achieve high accuracy. Consequently, many of these models are unable to run on easily deployable machines such as tablets.

1.2 Related Works

According to a paper published in 2019 by Quoc V. Le and Mingxing Tan from Cornell University, EfficientNet is a continuous family of models created by scaling each dimension with a fixed set of scaling coefficients. As a result, the depth, width and resolution of each variant of the EfficientNet models should be hand-picked to determine the best accuracy.

For instance, in a model used to classify Stanford Dogs, the model EfficientNet B0 was used. It has been shown in the study that transfer learning result is better for increased resolution if input images remain small. However, when training EfficientNet on smaller datasets, the model faces a risk of overfitting its data. Hence, data augmentation and pre-processing are important for EfficientNet. In other words, a useful tip is that in some cases, it might be beneficial to unfreeze only a portion of the layers rather than all, as this makes fine-tuning much faster when using larger models like B7. Another aspect to keep in mind is that larger variants of EfficientNet do no guarantee improved performance, especially for tasks with little data and few classes. In such a case, the larger variant of EfficientNet chosen, the harder it is to tune hyperparameters. In conclusion, it's important that the developers take time and experiment with all variants and play around with the layers in order to receive the best accuracy.

1.3 Problem Definition

The use of EfficientNet for waste classification may be ideal in public spaces where models with larger and more complex architectures are unable to run on small devices. Models such as ResNet scale up Convolutional Neural Networks (ConvNets) by adding more layers, and by scaling by depth [5]. However, it is not known if this is the most efficient scaling algorithm as previously, the process of scaling up ConvNets was poorly understood. This creates a problem in situations where models require a smaller size, yet still require high accuracy.

EfficientNet uses a new method of scaling to achieve better accuracy and efficiency greater than most traditional ConvNets [5]. Unlike conventional approaches to model scaling, where network dimensions are arbitrarily scaled, EfficientNet scales each dimension with a fixed set of scaling coefficients [5]. This results in a higher level of accuracy and efficiency. Furthermore, EfficientNet has several different versions along with EfficientNetLite versions that are specifically designed to run on mobile devices [5].

For waste classification, it is necessary for a model to classify waste quickly and accurately. Furthermore, when deployed in a public space, it is also necessary for the model to run on devices with limited storage capacity. Due to the high efficiency and accuracy of the EfficientNet, it is an optimal computer vision model to retrain for the purpose of waste classification.

2. METHODOLOGY

2.1 Training Data

The goal of our model is to successfully classify common waste items into four different categories. These categories are blue recycling (glass, plastic, and metal), grey recycling (paper and cardboard), along with organics, and landfill. These were based off the sorting categories in Kingston Ontario, as this is the preliminary location the model will be deployed. Using numerous public databases online, a collection of 4637 images was established for training data, summarized in Table 1. The quantity of each category was modified over time to reflect the difficulty the model had of classifying that category.

Table 1: Summary of Training Data

Category	**Image Quantity**
Blue Recycling	889
Landfill	1046
Organic	2301
Grey Recycling	404
Total	**4637**

2.2 Model Framework

To maximize the accuracy of the model, extensive research was conducted to determine the most appropriate framework. EfficientNet was found to be the most appropriate for this application as it can achieve high accuracy on the ImageNet dataset, while minimizing the number of parameters. This is very important for this application as the model will be run on an android tablet with limited computing power and must be capable of classifying an image in under a second. EfficientNet models between B0 and B5 were tested on our data, and it was found that the increase in input size from B0 to B3 caused significant improvements in accuracy, but further scaling had limited improvements. It was therefore determined that using the EfficientNet B3 framework using pretrained weights from the ImageNet dataset was the most appropriate baseline model. To tailor the model to this application, an average pooling layer, along with batch normalization, dropout, and fully connected layers

were added on top of EfficientNet. A summary of the current model can be seen in Figure 1.

Layer (type)	Output Shape	Param #
efficientnetb3 (Functional)	(None, 10, 10, 1536)	10783535
global_average_pooling2d (Gl	(None, 1536)	0
batch_normalization (BatchNo	(None, 1536)	6144
dropout (Dropout)	(None, 1536)	0
dense (Dense)	(None, 768)	1180416
dense_1 (Dense)	(None, 4)	3076

Total params: 11,973,171
Trainable params: 1,186,564
Non-trainable params: 10,786,607

Figure 1: Framework of image classification model

2.3 Training

The model was then trained for 20 epochs, until the validation set accuracy plateaued. to increase the size of the training dataset, data augmentation was used including rotating, zooming, shifting and flipping the preliminary training images. The model was trained in mini batches of 128 images, and learning rate decay was used to maximize the validation set accuracy. Before training, 10% of the data was set aside as the validation set which was used to measure the progress of the training. The accuracy and loss function of the training and validation sets are seen below in Figure 2.

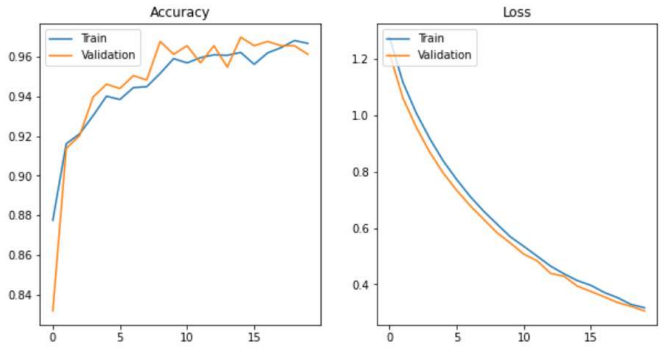


Figure 2 Accuracy and Loss function of training and validation data

As the training and validation set are not entirely representative of actual images seen by the model was deployed, a test set was developed to evaluate the model more accurately. Each of the four team members took approximately 100 images of common waste items around their home, in situations more representative of what the model will be expected to classify. These were then run through the model to predict how accurately the system would perform when deployed.

3. RESULTS AND DISCUSSION

By training the model shown above for 20 epochs, a peak validation set accuracy of 97% was achieved. The predicted categories compared to the true categories of the validation set for each of the categories is visualized in the confusion matrix in Figure 3.

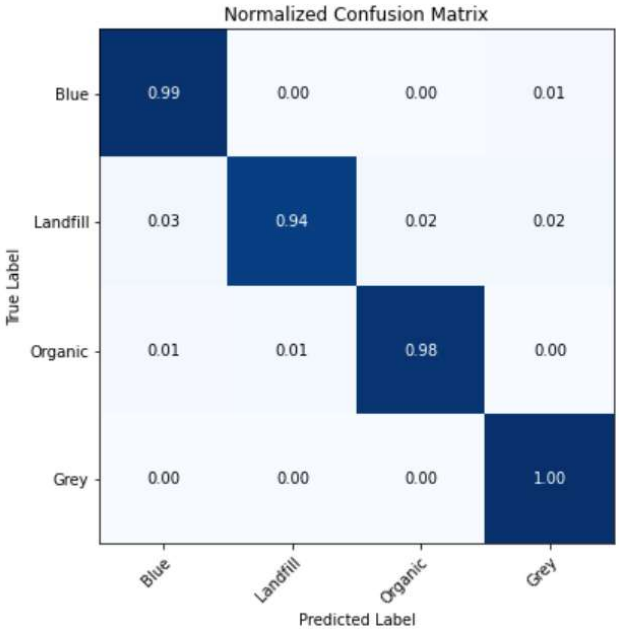


Figure 3: Normalized confusion matrix of validation set

The testing data collected by the team was then run through the model, producing an accuracy of 80%. This is likely representative of the accuracy the model achieves when deployed around campus, indicating that there is still further work to be done. The predicted categories compared to the true categories of the test set for each of the categories is visualized in the confusion matrix in Figure 4.

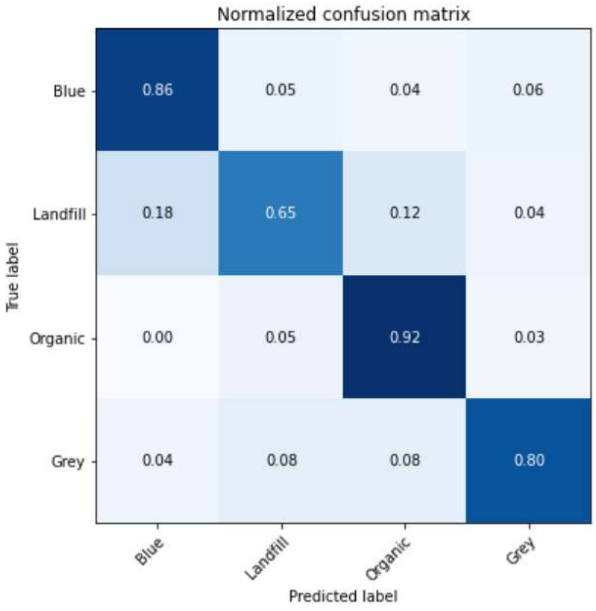


Figure 4: Normalized confusion matrix of test set

Testing revealed that there was a significant drop in accuracy between the validation set and test set. This means that the data used in the training and validation sets are not entirely representative of the testing set, which explains the drop in accuracy. When examining the training data, the majority of the photos are taken with white backgrounds, good lighting, and with the object taking up the majority of the frame. This is not the case with the testing data where the background is often a significant part of the image, taken in less than ideal lighting conditions. To improve the performance of the model on representative images, the training set would need to be further expanded, or the quality of the representative images would need to be improved by reducing background and lighting effects.

4. CONCLUSIONS AND FUTURE WORK

Recycling can be too easily contaminated when people do not ensure their waste is placed in the proper bin. This can be extremely damaging to the recycling initiatives and efficiencies. This project has accomplished the training and deployment of a convolutional neural network to properly classify waste items into their respective categories: blue recycling, grey recycling, landfill, and organic. This was done using transfer learning from the EfficientNet model which has been converted into a TensorflowLite model to be deployed locally on an Android tablet. This model will be used on Queen's campus to help Queen's students recycle more responsibly.

The model can properly classify common waste items with a 98% accuracy on the validation set. Even with background noise, activity in the background of images, that is commonly found in realistic deployment of software such as this one, the model is still able to perform with 80% accuracy.

Currently the development on the Android application is ongoing. Although the model is completely functional in the Android application the user experience is still being improved for ease of use to Queen's students. Along with improved UX additional resources are being implemented into the app so that in conjunction with the model the team can ensure students have assets easily accessible to completely responsibly dispose of their waste items. One of the major additions to the application is common exceptions with the waste disposal instructions. On campus certain products are specifically designed to be compostable even though visually entire plastic and similar cases. This is being done with location specific items so that managers of this software can easily set the location of the tablet to provide location specific instructions and suggestions.

Future steps also include the secure installation of Android tablets. To ensure the security of the tablets they are being installed with brackets at the most popular locations on campus. This way the model can be delivered with ease of use and peace of mind from any vandalism or theft.

REFERENCES

[1] D. Hoornweg, P. Bhada Tata, "What a Waste: A Global Review of Solid Waste Management, World Bank, 2012.

[2] GFL Environmental, "2016 Waste Audit Report", Queen's University, October 2016.

[3] Y. Chu, C. Huang, X. Xie, B. Tan, S. Kamal, X. Xiong, "Multilayer Hybrid Deep-Learning Method for Waste Classification and Recycling", Computational Intelligence and Neuroscience, 2018.

[4] V. Ruiz, A. Sanchez, J.F. Velez, B. Raducanu, "Automatic Image-Based Waste Classification", Bioinspired Systems and Biomedical Applications to Machine Learning, 2019.

[5] M. Tan, Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", International Conference on Machine Learning, 2019.

Predictive Diabetic Risk Modeling

David Huang[1], Nick Cheney[2], Susan Kovarik[3], Ellie Mehltretter[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: 17dh11@queensu.ca
2 e-mail: 16nrc2@queensu.ca
3 e-mail: 17slk6@queensu.ca
4 e-mail: ellie.mehltretter@queensu.ca

Abstract: As diabetes prevalence continues to accelerate globally, methods to better monitor diabetes progression are critical in enabling effective preventative action and reducing the burden on local healthcare systems. Current prognostic models often prioritize between interpretability, by way of stratification on a small set of lab values, and predictive accuracy using deep learning methods on high dimensional data. The objective of this study is to develop a wholistic diabetes risk model that has strong predictive ability and maintains interpretability. Recurrent neural network models were developed on rich EMR data to predict the onset of 10 diabetes-related complications and time series forecasting of six clinically relevant lab tests was used for risk stratification following American Diabetes Association clinical guidelines. We achieved AUC scores greater than 85 for 5 out of ten complication onset models, while lab value forecasting for risk stratification using LSTM and ARIMA models achieved satisfactory RMSE values. In combination, our models provide a comprehensive understanding of the relative risk level for individuals with diabetes.

1. INTRODUCTION

1.1 *Motivation*

The global diabetes burden is expected to increase from 463 million people in 2019 to 578 million people by 2030 with developed countries seeing the greatest increase in prevalence rates [1]. In Canada, diabetes prevalence is expected to increase from 11,232,300 in 2020 to 13.6 million or 32% of all Canadians by 2030. Moreover, the increase in diabetes prevalence presents a significant burden on the health-care system. With the direct cost to the Canadian healthcare system expected to increase from 3.8 billion in 2020 to 4.9 billion by 2030 [2].

Thus, it is critical to develop improved monitoring methods to track the overall health status of those living with diabetes to reduce the diabetes burden on the healthcare system and to ensure preventative action can occur before development of life-threatening complications.

1.2 *Related Works*

Due to the complex and diverse pathophysiology of diabetes, the American Diabetes Association (ADA) recommends individualized treatment and medication plans [3]. As such several studies have focused on personalizing treatment by scoring, or stratifying, the relative health of diabetic patients using clinical test values. These stratification methods allow for better resource allocation, help clinicians better monitor the relative health of their patients and have shown to improve overall diabetes outcomes [4].

More recently, several prognostic machine learning models have been developed alongside the increased adoption of electronic medical records (EMR) systems by healthcare providers. Excellent in finding statistical patterns in rich data, Ljubic et. al. demonstrated the potential for deep learning models trained on EMR data for Alzheimer's onset prediction. To capture the richness of EMR data they trained separate LSTM models on diagnoses, lab tests, and drug domains. The drug and lab test domains produced the best results

with 0.985 and 0.986 AUPRC respectively while the diagnoses domain achieved 0.651 AUPRC [5].

1.3 *Problem Definition*

The complex nature of diabetes and related complications make it difficult to quantify a patient's diabetic risk level. Earlier approaches attempt to quantify risk by stratifying diabetic patients on a small subset of well-controlled clinically relevant lab tests. While stratification by this means is well adopted due to increased interpretability and practicality in a clinical environment, it severely under-utilizes the wealth of data available in today's EMRs.

On the other hand, recent deep learning models have taken advantage of high-dimensional data sources and have shown high prediction accuracy for prognostic disease models. However, due to the 'black box' nature of deep learning models and a subsequent lack of interpretability, deep learning models have not seen wide-spread adoption in a clinical setting.

Our goal is to develop a wholistic risk model for diabetic patients that monitors and predicts their overall diabetic health using diabetes stratification methods while also predicting the onset of diabetes related complications using high accuracy machine learning models.

2. METHODOLOGY

2.1 *Data*

To create a comprehensive risk profile for diabetic patients, we developed two time-series models on EMR data from the Canadian Primary Care Sentinel Surveillance Network (CPCSSN). The CPCSSN database is comprised of anonymized clinical information from patients presenting with a wide variety of diseases and is split into several domains such as billing information, patient demographics, lab tests and medication [6]. Our first model used the lab and exam domains to forecast the values of 6 clinically relevant lab tests while our second model used billing, demographic, lab, and exam domains to predict the onset of diabetes-related complications. Diagnosis' codes were found in the billing domain and were represented using International Classification of Diseases-Ninth Revision codes (ICD9).

2.2 *Diabetes Stratification*

Following the American Diabetes Association (ADA) clinical guidelines [3], stratification levels were calculated for HbA1C, blood pressure (systolic and diastolic), high- and low-density lipoproteins and triglycerides, and albumin/creatinine ratio lab results. Two threshold values for each lab test determined the relative stratification level, 1 to 3, of patients where a score of 1 represented low values, 2 represented normal values, and 3 represented high values. The scores for HbA1C, blood pressure, lipids, albumin/creatinine ratios and multi-category lipid averages were found and summed to generate a final risk score.

Given the time series nature of EMR data, ARIMA and RNN models were used to forecast the selected lab values. The ARIMA model is a widely used statistical method for analyzing time series data. The model takes three parameters p,d,q where p is the order of the AR (autoregressive) term or lag order, d is the differencing order, and q is the order of the MA (moving average) term respectively. We used the standard parameters of $p = 5$, $d = 1$, $m = 0$ for our analysis.

Outliers from the series were removed by only taking data that fell within the inter-quartile range for each respective feature and 14-day windows were used to produce fixed time-series data. Thus 14-day forecasts were generated for each feature and re-stratification could occur using ADA threshold values. An 80/20 split was used for the LSTM model and each model was trained on an average of 800 samples.

2.3 *Complication Prediction*

In our second approach we sought to independently predict the onset of 10 diabetes-related complications. These complications were angina pectoris, atherosclerosis, ischemic heart disease, depressive disorder, diabetic nephropathy, diabetic neuropathy, diabetic retinopathy, hearing loss, myocardial infarction, and peripheral vascular disease. This was done using both patient diagnosis data and combined lab and exam result data. These sources of information provided two approaches which were developed separately, with the goal of consolidating the results to form a single model with the highest accuracy. In both approaches, two deep learning models were employed consisting of recurrent neural network (RNN) unidirectional LSTM and bidirectional RNN gated recurrent unit (GRU) architectures. The complication

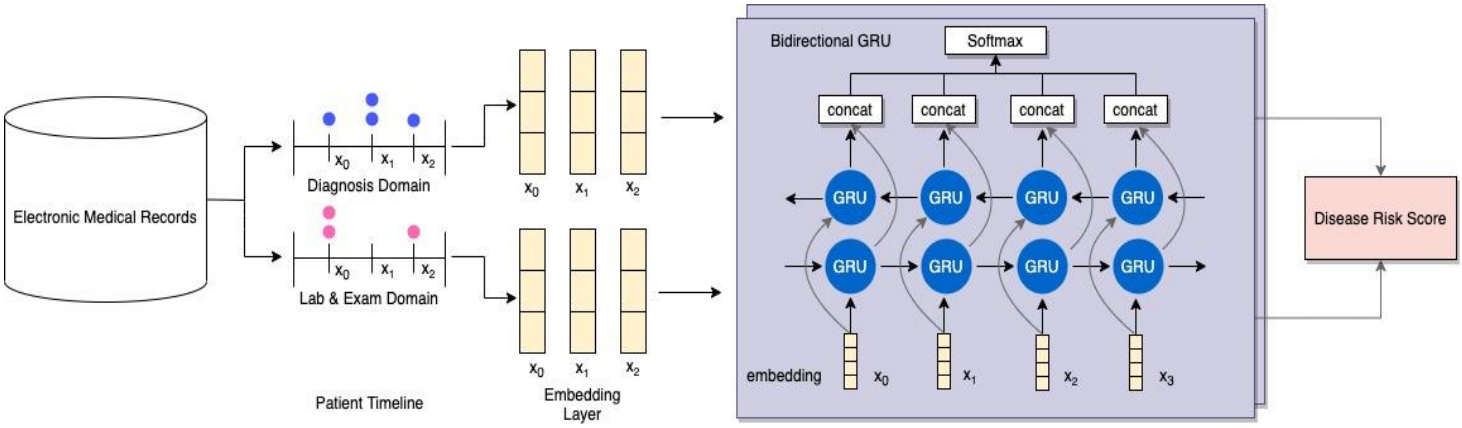


Figure 1: Onset of complication model diagram consisting of: the patient timeline as an input layer, an embedding layer, bidirectional GRU model architecture, and a disease risk score as the output layer.

diagnosis data points were ascertained from the billing table via recorded ICD-9 code ranges unique to each complication. A negative dataset was also created for each positive complication dataset by selecting age and gender matched diabetic patients without complications. Data were filtered to exclude results after the date of complication onset to prevent data leak and patients in both positive and negative datasets excluded results prior to each patient's date of diabetes onset. Additionally, patients were only included if they had at least 4 visits and at most 51 visits.

For the diagnosis domain, each patient's data were transformed into a one-hot encoded matrix of dimensions m x n, where m represented the number of rows or unique dates with at least one result and n = ***3397*** represented the full range of possible ICD-9 codes that presented at least 10 times in the data. Thus, each matrix element e_{ij} had a value of 1 if the patient had a diagnosis with an ICD-9 code *j* on a given date *i*, and a 0 otherwise.

For the lab and exam domains, the test results were filtered to include only 18 selected features, and outliers for each test type were removed. Each patient's results were again represented as one-hot encoded matrices by binning each test using the 33rd and 67th percentile values for each respective feature as thresholds resulting in m x n matrices. Again, m represented the number of unique dates with one or more lab/exam results, and n = 54 represented low, medium, high bins for each feature (18 x 3).

In both approaches, patient's with less than 50 visits were padded with zero vectors until they had 50 rows. Singular value decomposition (SVD) was used to reduce the dimensionality of input matrices, resulting in a final dimensionality of 50 x 50 or 2500 features per patient for the diagnosis domain and 18 x 50 or 900 features per patient for the lab and exam domain. Finally, encoded patient timelines were modeled using LSTM and bidirectional GRU layers. Softmax activation was used to generate onset probabilities. Figure 1 shows the complication model diagram. A 90/10 training/testing split was used and accuracy was evaluated using area under the receiver-operator curve (AUC) metric for each complication and approach.

3. RESULTS AND DISCUSSION

The 5-1-0 ARIMA model produced good results with the systolic and diastolic blood pressure data. For the diastolic blood pressure model, a root mean squared error of 3.620 mmHg was achieved on a range of diastolic blood pressure values between 82.0 mmHg and 71.0 mmHg. For the systolic blood pressure data, a root mean squared error of 6.264 mmHg was achieved on a range of values between 139.0 mmHg and 121.0 mmHg. The LSTM model produced the best results for HbA1C, HDL, LDL, triglycerides, and albumin/creatinine ratio with RMSE values of 5.3 (%), 14.2 (mmol/L), 13.5 (mmol/L), 50.3 (mmol/L), 32.1 respectively.

For our diabetes-complication onset models, it was found that using the bidirectional GRU model architecture yielded higher AUC and accuracy values than LSTM models in all cases, leading us to use it primarily for model evaluation. A model was separately constructed and evaluated for each complication and data source, with the exception of the diabetic retinopathy model using lab and exam data, which lacked a sufficient positive sample size (n < 1000). The results are summarized in Table 2.

The lab and exam data domain produced higher accuracies for seven out of the ten complications than the diagnosis domain, suggesting that this source of data was a better choice for our solution. This finding is also consistent with previous works [5]. We also found that models using lab and exam data with less than 1000 positive patients performed worse with AUC scores less than 0.8. This is not a surprising finding since many deep learning models typically require large training sets for higher accuracies.

Complication	*Diagnosis Data AUC*	*Lab/Exam Data AUC*
Angina pectoris	0.6373	0.9463
Atherosclerosis	0.6932	0.6249
ICHD	0.6941	0.9273
Depressive disorder	0.6714	0.8122
Nephropathy	0.6865	0.9108
Neuropathy	0.6970	0.9221
Retinopathy	0.6824	N/A
Hearing loss	0.7121	0.6659
MI	0.6298	0.8701
PVD	0.5098	0.5201

Table 1: Results from RNN Bidirectional GRU Models trained on the diagnosis and lab and exam domains for complication prediction; ICHD: ischemic chronic heart disease; MI: Myocardial infarction; PVD: Peripheral Vascular Disease.

4. CONCLUSIONS AND FUTURE WORK

As global diabetes prevalence continues to rise, diabetes monitoring and forecasting models are critical for early intervention and prevention of costly complications. In this study we set out to develop a wholistic diabetes risk model that encapsulates overall disease status and diabetes related complication development likelihood. We were able to successfully make a two-week prediction on six clinically relevant lab tests and subsequently calculate stratification levels as a general risk score. We expanded on this approach by developing separate GRU models for complication onset prediction with >0.85 AUC scores for 5 complications.

Without knowledge of the clinical domain, it is difficult to determine which features are of the most importance in determining diabetes risk levels. The features presented in this study, for both models, were found in literature review where data and features often differ from study to study. As such, a combination of features were selected based on frequency and relevancy reported by other researchers. To improve model accuracy, better feature selection can be achieved with the help of a clinical consultant. Specifically, features with high predictive value such as cholesterol or Alkaline Phosphatase in serum were excluded due to a lack of references to these features in the literature.

In addition, combining the diabetes related complication models trained on diagnoses data and lab data has been shown to improve overall model accuracy [5]. An ensemble model is proposed where model inputs would include complication onset probabilities from individual GRU models. Moreover, an ensemble model allows for greater domain usage as patient demographic data such as age, gender, and risk factors can be included as model inputs, providing even greater predictability and interpretability.

REFERENCES

[1] S. Pouya, "Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the International Diabetes Federation Diabetes Atlas, 9th edition," *Diabetes Research and Clinical Practice,* vol. 157, no. 107843, 2019.

[2] Diabetes Canada, "Report on Diabetes – Driving Change," Ottawa, 2015.

[3] American Diabetes Association, "Clinical Practise Recommendations," *Diabetes Care,* vol. 22, 1999.

[4] J. J. W. S. R. L. M. L. M. S. R. C. G. P. Charles M. Clark, " A Systematic Approach to Risk Stratification and Intervention Within a Managed Care Environment Improves Diabetes Outcomes and Patient Satisfaction," *Diabetes Care,* vol. 24, no. 6, pp. 1079-1086, 2001.

[5] S. R. X. H. C. M. P. S. O. R. N. L. G. Z. O. Branimir Ljubic, "Influence of medical domain knowledge on deep learning for Alzheimer's disease prediction," *Computer methods and Programs in Biomedicine,* vol. 197, 2020.

[6] A. L.-L. K. M. J. A. L. R. M. S. K. R. B. Tyler Williamson, "Primary Health Care Intelligence," Canadian Primary Care Sentinel Surveillance Network, 2013.

Autonomous Highway Simulation with Smart Cruise Control

Andrew Simonds[1], Dylan Moss[2], Brock MacDonald[3], Azeem Quadri[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: andrew.simonds@queensu.ca
2 e-mail: 15dm28@queensu.ca
3 e-mail: 16bnm5@queensu.ca
4 e-mail: 16saaq@queensu.ca

Abstract: *As major automotive companies continue to incorporate new forms of automation into their fleet of vehicles, one of the major tasks that remains is perfecting self-driving capabilities on highways. The goal of this project is to use a TurtleBot and the sensors that are provided on the robot to simulate various aspects of an autonomous highway. We used Gazebo to facilitate testing on the TurtleBot as we developed a multitude of scripts to recognize lanes, adjust steering and speed, scan for other robots, and remain in the center of the lane. The robot is also able to recognize if there is another robot in its field of view through both a laser scan and an image recognition model to match the speed of another robot in front of it.*

1. INTRODUCTION

1.1 Motivation

It is evident that our society is currently on the brink of the next transportation revolution, where autonomous vehicles are becoming more prevalent on the roads. Due to this increasing popularity, high-quality data sets and continuously evolving algorithms are steadily improving the frameworks of autonomous driving. These systems are optimized in the hopes to reduce crashes, pollution and traffic jams while making autonomous transportation attainable by all. Given this exponential rise in autonomous driving and considering the numerous benefits of these systems, our team found autonomous highway driving to be a compelling topic to study.

1.2 Related Works

To achieve smart cruise control as a step towards autonomous driving the first step is to detect the lanes on the road. The traditional approach is to use computer vision techniques without machine learning. This approach as described in *"A Precise Lane Detection Algorithm Based on Top View Image Transformation and Least-Square Approaches"* [1] involves a perspective transform followed by applying edge detection techniques such as Canny Edge Detection and Hough Transform to detect the lane pixels. Next, a polynomial fitting technique is performed on the lane pixels to create a single representation for each lane.

An alternative approach that has now become standard in industry is to apply deep learning techniques to the problem. As described in *"Reliable multilane detection and classification by utilizing CNN as a regression network"* [2] this method has been found to be more reliable and robust across a range of conditions and does not require as much fine-tuning to the individual set-up. The trade-off of this method is it requires much higher computational power to achieve usable results.

The next step in the smart cruise control pipeline is to create a steering algorithm using the detected lanes as input. In "*Robust PID Steering Control in Parameter Space for Highly Automated Driving"* [3] a methodology for designing and tuning the controller used for the vehicle steering is laid out to maximize stability. A similar methodology can also be adapted for the vehicle's acceleration using the detected distance towards the next vehicle as input.

The final feature required for smart cruise control is the detection of other vehicles and objects. This can be used to customize the action of the steering and acceleration controllers based on the type of detected object or be used to initiate other actions such as lane changes. In "*Vehicle Detection in Deep Learning"* [4] both the traditional computer vision and deep learning techniques are compared with the conclusion being a CNN can achieve much more accurate results.

1.3 Problem Definition

Following research into related works in Section 1.2, the team came to a consensus that a relevant problem to solve would be an autonomous vehicle in a highway simulation environment with regulated steering and speed based on external factors. We believe that many automotive companies in industry are just now breaking into this domain, so the research being done into this area will provide another great opportunity for improvements in the field of autonomous vehicles.

2. METHODOLOGY

2.1 Preprocessing

The team used a simulated TurtleBot in Gazebo to run our models and algorithms. Using ROS2 as our programming framework, we were able to extract integral information from the various sensors on the TurtleBot such as velocities and position. It was then possible to publish to these same topics to meet the needs of the project. The input data came from the robot's LIDAR scanner, the camera feed, and the velocity sensors. The LIDAR scanner provided distance measures for potential obstacles at every degree around the robot, but the search was narrowed to a smaller field of view in front of the robot to scan for certain obstacles. The camera data was manipulated through various image processing techniques to extract lane information, along with being saved to use for robot detection. Lastly, the velocity data was required to ensure the robot moved at the correct speeds and was able to properly adjust for either turns or obstacles in its path.

2.2 Solution Implementation

Once all the data was easily accessible through the multitude of ROS topics, the team worked towards the initial goal of accurate lane detection and subsequent robot reaction. Initially, two approaches were taken to tackle staying within the lanes while traversing the track. The first was to analyze the angles of the lanes made in the camera feed and try to optimize the difference between the two to be zero, however this was dropped in favor of the more reliable method of centering the robot between the two base pixels of the detected lane. The reaction of the robot was to optimize its angular velocity to remain centered in the lane using a Proportional-Integral-Derivative (PID) controller [5]. Next, the team used the LIDAR data coupled with an image classification model to implement a smart cruise control feature. Using the LIDAR distance data and a PID controller, a TurtleBot is able to maintain a predetermined linear distance from any object in its path. To ensure the robot is only maintaining a distance from another TurtleBot and not something such as a pedestrian or a tree, this feature is backed with an image classifier to determine if there is indeed another robot in front of it.

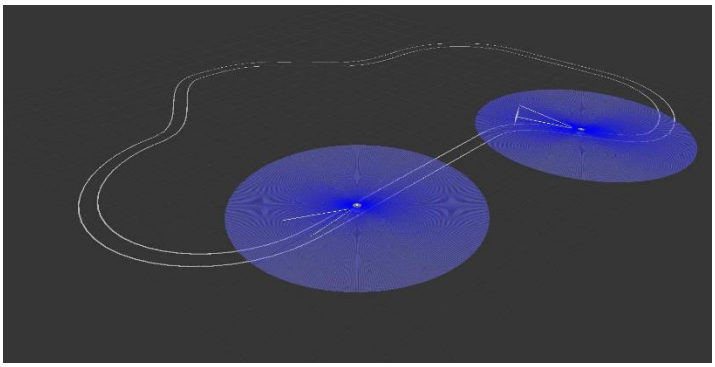


Figure 1: TurtleBots Driving Around Track in Gazebo.

3. RESULTS AND DISCUSSION

Using these detected lanes as input the PID steering controller worked sufficiently after several iterations of tuning. The TurtleBot could successfully navigate indefinitely around the track while stably remaining within the lanes. If the velocity was increased too high the TurtleBot experienced substantial overshoot on tight corners but was able to correct within a few seconds.

The LiDAR scan was also successful at detecting other TurtleBots and obstacles. Using this detected distance as input the acceleration controller successfully matched speeds to a leading TurtleBot with an alternating low and high velocity. The track consisted of a maximum of a forty-degree turn, and the system accurately handles the distance tracking for these turns.

For the CNN classifier, a data set consisting of 650 images for each of the two classes was collected within the Gazebo simulation. The data was split into a training set of 500 images and a test set of 150 images. On this data, the model achieved an accuracy of 99%. To validate these results the classifier was tested live in the Gazebo simulator. The system accurately detected the TurtleBot at a variety of locations and distances from the camera and successfully worked when integrated with the acceleration controller.

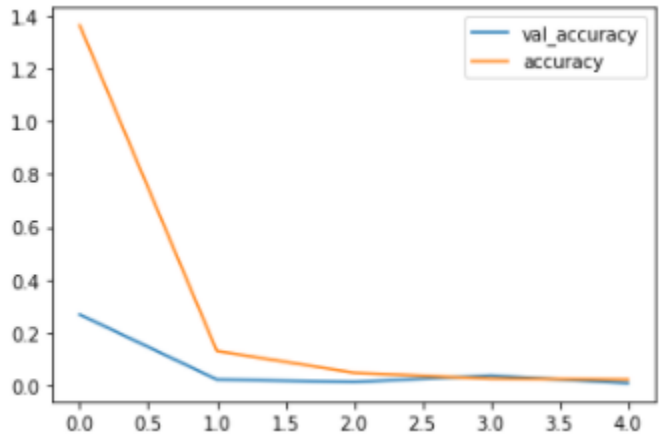


Figure 2: Model Loss versus Epochs trained.

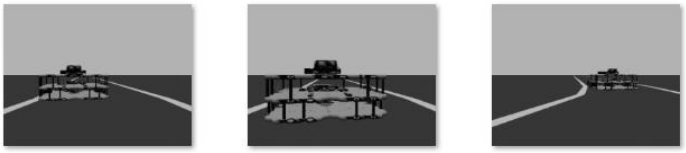


Figure 3: Examples of the TurtleBot Class.

Figure 4: Example of the Empty Class.

4. CONCLUSIONS AND FUTURE WORK

In conclusion, the team has utilized the simulation of a TurtleBot in Gazebo to construct the models and algorithms needed to simulate autonomous highway driving, with a smart cruise control system. Using a TurtleBot allowed for easy real-time data collection of essential information which allowed the team to have great success with the project overall.

Future work of this project will consist of additional testing and optimization of the current system, along with the implementation of features from the other Highway Simulation team which consist of lane detection on a two-lane highway, and lane changing functionality. With all the features compiled into one system, the team would then develop a passing algorithm to allow for one robot to change lanes and pass by a slow-moving robot in front of it. In addition to the new features, it is paramount to test the code on a physical TurtleBot to determine how certain sensors and actuators behave in a real environment.

REFERENCES

[1] B. Dorj and D. J. Lee, "A Precise Lane Detection Algorithm Based on Top View Image Transformation and Least-Square Approaches," 30 November 2015. [Online]. Available: https://www.hindawi.com/journals/js/2016/405809/. [Accessed 15 March 2021].

[2] S. Chougule, N. Koznek, I. A. G. Adam, V. Narayan and M. Schulze, "Reliable multilane detection and classification by utilizing CNN as a regression network," 2018. [Online]. Available: https://openaccess.thecvf.com/content_ECCVW_2018/papers/11133/Chougule_Reliable_multilane_detection_and_classification_by_utilizing_CNN_as_a_ECCVW_2018_paper.pdf. [Accessed 16 March 2021].

[3] M. T. Emirler, I. M. C. Uygan, B. A. Guvenc and L. Guvenc, "Robust PID Steering Control in Parameter Space for Highly Automated Driving," 4 February 2014. [Online]. Available: https://www.hindawi.com/journals/ijvt/2014/259465/. [Accessed 15 March 2021].

[4] A. L. Abbott, D. Pillis and R. M. Buehrer, "Vehicle Detection in Deep Learning," Virginia Polytechnic Institute and State University, Blacksburg, 2019.

[5] National Instruments Corp., "PID Theory Explained," 17 May 2020. [Online]. Available: https://www.ni.com/en-ca/innovations/white-papers/06/pid-theory-explained.html. [Accessed 16 March 2021].

This project can be found at:
https://github.com/andrewsimonds14/Highway-Simulation

Virtual Assistant Attention Detection

Ethan Callanan[1], Jae Makitalo[2], Ella Duffy[3], Kavin Zhu[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: e.callanan@queensu.ca
2 e-mail: jae.makitalo@queensu.ca
3 e-mail:17emld@queensu.ca
4 e-mail:18kz44@queensu.ca

Abstract: *The popularity of virtual assistants has been rising at an exponential rate, but they all use the same unnatural method of keyword activation. The goal of this project was to develop a novel system to provide a more natural interface for interacting with virtual assistant devices. To achieve this, we developed an attention detection system using a multitask cascaded convolutional neural network for face detection and a convolutional neural network for attention classification. The face detector performs with a true positive rate of 95.04%, and the attention classifier performs with 97.2% testing accuracy. The attention detection pipeline was implemented in a web application simulating a virtual assistant. We plan on improving the generalizability of the attention classifier by training it on a larger and more diverse dataset, and we plan on implementing the model in a dedicated device.*

1. INTRODUCTION

1.1 Motivation

The use of virtual assistants (VA) has seen a meteoric rise in past years. In the last two years alone, the number of VAs in use worldwide has risen from 3.25 billion to 4.2 billion. By 2024 that number is projected to overtake the world population with approximately 8.4 billion devices [1]. The text-to-speech recognition segment of the VA market alone was valued at USD 2.2 billion in 2019, and the market is expected to grow at a rate of 34.4% over 2020 to 2027 [2]. Despite the technology's incredible popularity, the way users interact with the devices has not seen any development. In social interactions, humans naturally focus their attention on the speaker; however, none of the major devices implement vision based interaction and instead opt for unnatural keyword activation

1.2 Related Works

Developers in the VA field have begun incorporating computer vision in their products for applications unrelated to activation. Google has implemented gesture controls and uses facial recognition for personalized display in their Nest Hub Max, and Amazon utilizes face detection to orient the Echo Show towards the user. Previous applications of attention detection have largely focused on driver monitoring. Although they are not designed for VAs, they operate on the same principles. The most notable implementation is Comma AI's driver monitoring system, which utilizes eye tracking and image classification to determine whether or not the driver is paying attention to the road. Researchers at the Massachusetts Institute of Technology (MIT) built a gaze estimation model for driver monitoring which avoids the use of eye tracking [3]. Instead, they opted to perform face detection with a Histogram of Oriented Gradients and linear support vector machine to detect faces, extract the facial landmarks with a cascade of regressors from a facial landmark mark-up, and classify the gaze direction in one of six regions with a random forest classifier.

1.3 Problem Definition

In recent years, the market and use of VAs has grown rapidly, but the way in which we interact with these assistants has been largely overlooked. We set out to design a novel method of interaction, using computer

vision, to provide a user experience that more closely resembles that of a normal conversation. As mentioned in *Section 1.2*, using computer vision for tasks related to attention detection has been explored before. CommAI's driver monitoring system heavily relies on eye tracking, however, eye detection is unreliable in this application due to the many angles and lighting conditions a user may interact with the device at. MIT proposes a more suitable implementation, however their system uses a six stage pipeline and is a six class classifier. For the purposes of VA activation, binary classification will suffice and is both simpler and less computationally expensive.

2. METHODOLOGY

2.1 Dataset Generation

Training and evaluation is carried out on a dataset of 10 subjects. For each subject, there are 200 real images and 324 synthetic images, providing 5240 total images [4]. The images are varied in illumination, background, and pose (by up to 30 degrees in either direction). This dataset was supplemented with an additional 200 images of a sitting subject with similar variety. Preprocessing of the images involved converting to grayscale and resizing to 224 pixels along the smallest edge (maintaining the aspect ratio). Each image was labelled as either "attentive" or "inattentive" based on whether or not the subject was looking towards the camera. A 20% test split was used to evaluate the models.

2.2 Solution

The solution consists of a two step pipeline: face detection and attention classification. If the system passes the first step (face detection) the attention classifier is activated and makes the binary decision as to whether or not the user is focusing their attention on the device.

2.3 Face Detection

A multitask cascaded convolutional neural network [5, 6] (MTCNN) was used to identify if a face is present in an image frame. The network consists of three stages in the form of independent convolutional neural networks (CNN).

The first stage, the proposal network, uses a fully convolutional network[1] (FCN). This network finds windows in the image that could potentially contain a face as bounding box regression vectors. The network performs some refinement to combine overlapping regions, and outputs the remaining candidate windows. Next, the refine network performs calibration with bounding box regression and uses non-maximum suppression to further combine overlapping windows. It then outputs whether each candidate contains a face or not, along with a bounding box and vector for facial landmark localization (eyes, nose, and mouth).Finally, the output network operates in a similar fashion to the refine network, but describes the face in more detail. This final stage outputs the binary face classification, along with the bounding box and five absolute landmark locations: the two eyes, nose, and mouth corners.

2.4 Attention Classification

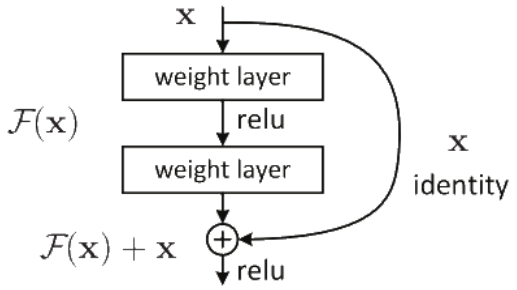


Figure 1: A residual block in the ResNet architecture. Layers can skip subsequent layers in the network through an identity shortcut connection.

A CNN was used to make a binary classification on the attentive state of a face. The classifier uses the ResNet [7] architecture with 50 convolutional layers. The network achieves far better results with less training than its shallower counterparts, and manages to avoid the problem of vanishing gradients[2] by introducing identity shortcut connections. These connections allow a layer to skip the subsequent layers and map its output directly to a layer further in the network as shown in figure 1. The first layer is a 7×7 kernel, the second layer is a 3×3 max pool, and each subsequent layer is a 3×3 kernel, all using rectified linear unit activation. Dropout was applied for regularization and to prevent co-adaptation of neurons.

[1] A CNN without a dense layer.

[2] Repeated multiplication during backpropagation causes the gradient to shrink. If a network is sufficiently deep, this will cause massive degradation in performance.

Training was performed with the Adam optimizer [8] using negative log likelihood loss for 10 epochs.

2.5 Virtual Assistant Integration

The model was implemented in a Streamlit web-application made to simulate a VA device. The model analyzes every other frame to make its classification on the user's attentive state. When the user is attentive for 10 consecutive frames (five consecutive positive classifications from the model), the app waits for the user to begin speaking and listens until they complete their sentence. The recording is then sent to a custom Dialogflow agent through the dialogflow API and both the audio and text responses are displayed to the user.

3. RESULTS AND DISCUSSION

The face detector performs at a true positive rate of 95.04% and the attention classifier achieved an accuracy of 97.2% on the test set.

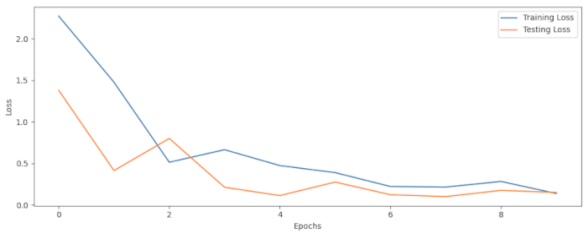


Figure 2: Learning curve of the attention classifier.

The accuracy of the models closely reflects its practical performance in conditions similar to the training data. Interacting with the assistant was a nearly seamless user experience[3], and false positives were handled by the assistant activation logic in the application. False negatives from the classifier occasionally delay the activation of the assistant, but these occurrences are infrequent enough not to diminish the overall user experience.

Despite the attention classifier's accuracy in controlled conditions, when presented with poor lighting or unfamiliar angles the performance suffered. This is likely largely due to the consistent set of lighting and angles in the training images. Additionally, all the images were taken at similar distances from the camera. As such, in significantly unfamiliar conditions the model will get stuck on one of the two classifications.

4. CONCLUSIONS AND FUTURE WORK

The two components of the attention detection pipeline were successfully built. Both models performed well in a testing environment and in controlled live environments. In unfamiliar contexts the attention classifier did not perform as well. Reflection on the training data suggests this was due to insufficient variety in the image attributes. The models were integrated with a proof of concept VA application and provided a positive user experience.

We aim to improve the generalizability of the attention classifier by training on a more varied dataset. Images of subjects taken from different angles, elevations, and distances will help the model handle the many edge cases that arise from live classification. Performance in poor light conditions may also be improved by adding more images in low light and with different light sources. Finally, we plan on implementing the model and activation logic in a dedicated VA device.

REFERENCES

[1] Statista Research Department, "Number of voice assistants in use worldwide 2019-2024", Statista, 22 Jan 2021.

[2] Grand View Research, "Intelligent Virtual Assistant Market Size, Share & Trends Report", Grand View Research, Apr 2020.

[3] L. Fridman, P. Langhans, J. Lee, B. Reimer, "Driver Gaze Region Estimation Without Using Eye Movement", MIT, 1 Mar 2016.

[4] "Face Recognition Database", MIT Center for Biological and Computational Learning.

[5] K. Zhang, Z. Zhang, Z. Li, Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks", IEEE Signal Processing Letters, 11 Apr 2016.

[6] https://github.com/davidsandberg/facenet

[7] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[8] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

[3] Demo interaction is shown at https://www.youtube.com/watch?v=0-YFEVMPsV8

Music Genre Classification Using K-Nearest Neighbours And Neural Networks

Caio Coelho[1], Renay Rahman[2], Benjamin Tilden[3], Walter Hasick[4]

Western AI Club
Western University, London, ON N6A 3K7, Canada.

1 e-mail: ccoelho6@uwo.ca
2 e-mail: rrahma33@uwo.ca
3 e-mail: btilden@uwo.ca
4 e-mail: whasick@uwo.ca

Abstract: *As one of the teams in Western AI's Intermediate Project cohort, we explored and learned about the use of supervised learning within music genre classification. Our team had an opportunity to extract song features, build and train multiple models, then compare and build a single, most accurate model as a team. Initially, we created a KNN model with an accuracy of roughly 45%. Then, we tried a simple neural network model and achieved 62% accuracy. Using the neural network approach, we created a final, multi-layer model that reached 66% accuracy. Our team believes the AI community can benefit from making models more accessible to external users. We often hear about new, advanced models being published, but the average person rarely gets to try them. For that reason, we decided to publish our model to a website that we created leveraging Flask and Tensorflow.js. Anyone can access the website, easily upload a song and get a prediction back from the model.*

1. INTRODUCTION

1.1 Motivation

With the increased popularity of music streaming services such as Spotify and Apple Music, the problem of music genre classification is becoming more relevant. The website "Every Noise at Once" reports 5,283 different genre distinctions on Spotify as of 2021-03-22 [1], which demonstrates the subjective nature of this classification and highlights a possibility of using Machine Learning to achieve a more accurate categorization.

This paper aims to compare the performance of K-Nearest Neighbors (KNN) algorithms to that of Artificial Neural Networks (ANN) in classifying songs into ten genres by analyzing Mel-Frequency Cepstral Coefficients (MFCC) extracted from songs. As part of this project, we were also motivated to make our classifier model easily accessible online. For that reason, we published a website [2] where anyone can upload a song and get back a classification for it.

1.2 Related Works

An essential part of our work was extracting MFCCs from the music files. The paper "Music genre classification using MIDI and audio features." [3] explores a similar approach to our problem and provides evidence that the accuracy of genre classification using MFCC is one of the best when compared to other feature extraction techniques such as BEAT, STFT, and MPITCH.

Similarly, the paper "Automatic Music Genre Classification for Indian Music"[4] provided insight on the applicability of KNN models using MFCC data but concluded that KNN is not the best model for the task at hand and that MFCC data alone is not enough to achieve the highest possible accuracy.

1.3 Problem Definition

The scope of this project is to explore how accurately simple KNN and neural network models can classify songs into genres. To perform this classification, we also had to explore ways to extract audio features from the songs using MFCCs; an audio file contains

millions of bits, so it cannot be easily used as input and has to be reduced to only a few coefficients.

2. METHODOLOGY

Instead of using a pre-existing dataset, our team elected to extract our data from songs to encompass the entire data science process further. Feature extraction is a necessary and crucial part of analyzing and finding relations between different songs. Raw audio data cannot be understood by models and instead must be converted into understandable format features, usually in the form of coefficients or single values.

First, raw audio data needed to be converted into a spectrogram. “A spectrogram is a visual representation of the spectrum of frequencies of sound or other signals as they vary with time.” [5] It’s a representation of frequencies changing with respect to time for given music signals [6].

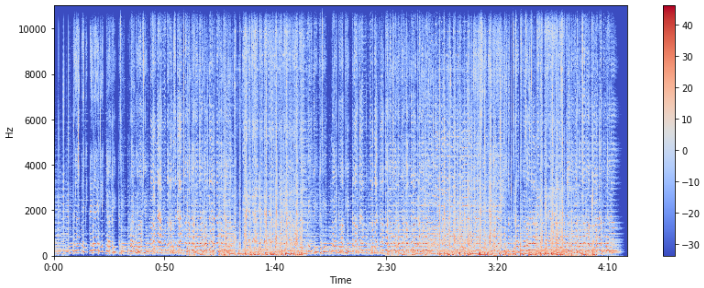


Figure 1: An example of a spectrogram

Upon converting the audio files, we were able to extract several features using a Python package, Librosa. Features included zero-crossing rate, spectral centroid, spectral roll-off, and most importantly (the features which had the most significant effect on the accuracy of our model), the Mel-Frequency Cepstral Coefficients. MFCCs of a signal are a small set of features (usually about 10–20) that concisely describe the overall shape of a spectral envelope [6]. Once we extracted the features of all 1000 songs used in our training data, we were able to classify songs into different genres.

To classify the datasets, the group elected to begin with a KNN algorithm. KNN was selected as a starting point because the implementation is relatively simple, and it requires very little computational complexity. The results of the KNN model were then used as a baseline for evaluation of more complex models better suited to the task of classifying music genres. As the KNN model is sensitive to noisy data and outliers, the optimal model would be robust enough to handle outliers and predict more complex non-linear relationships. For these reasons, the group elected to proceed with an Artificial Neural Network (ANN) model. To evaluate the proposed solution, it would have to achieve a higher accuracy than the KNN model on the test set.

Additional analysis was performed on the ANN model to determine the optimal number of hidden layers, in addition to hyper parameter tuning. To determine the ideal number of hidden layers, the hyper parameters were held constant while adding layers until the system's performance increase did not outweigh the computational cost. Once the number of hidden layers was selected, the hyper parameters such as learning rate were tuned until the model achieved the highest possible accuracy.

3. RESULTS AND DISCUSSION

To determine if the ANN model was appropriate for the task of classifying music genres, it was compared against the baseline KNN model. Table 1 demonstrates the results of each model.

Model	*Test Accuracy*	*Train Accuracy*
KNN	42.88%	61.05 %
ANN_1	63.80 %	96.38 %
ANN_2	64.90 %	98.62 %
ANN_3	65.80 %	99.37 %

Table 1: Accuracy Results of models

ANN_1 describes the most basic ANN model with two hidden layers, while ANN_2 describes a 3 hidden layer model with no hyperparameter tuning, and finally ANN_3 represents the third and most robust model containing 3 hidden layers in addition to hyperparameter tuning. It is evident from the results that the ANN is indeed an appropriate model to classify the data as its performance is superior to that of the KNN. Additionally, it is apparent that the accuracy increases as the model complexity increases, which is an important insight. Figure 2 shows the

training and validation accuracy as a function of number of epochs

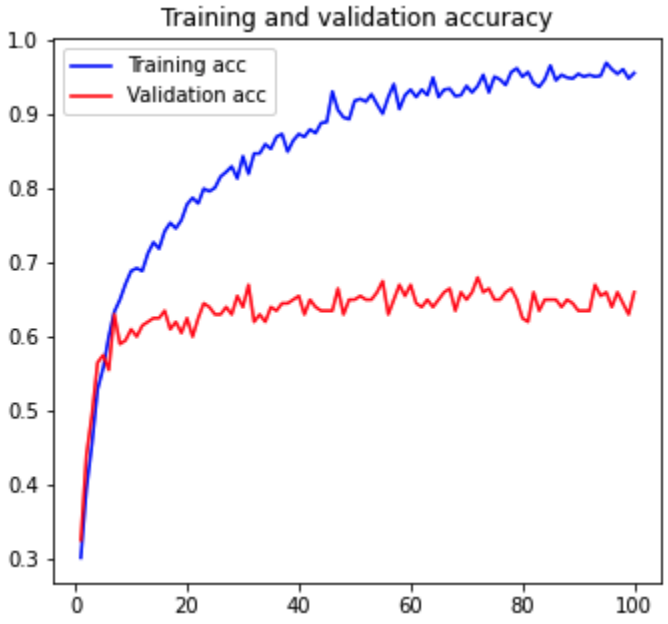


Figure 2: Training and validation accuracy plot

The result of Figure 2 again demonstrates the correlation between increase in hyper parameters, and improved accuracy in the model. The results are as expected and validate the assumption that increased model complexity will lead to increased performance.

From the project, the group has learned that model selection contributed the most to the results. Although slight increments in performance were achievable using hyperparameter tuning, having the right model for the task is the best approach to improve the performance.

4. CONCLUSIONS AND FUTURE WORK

All in all, our project produced a model that correctly classifies songs into genres with 65.80% accuracy. During the development period, our team created models that use extracted MFCCs to categorize the songs. The first model, using a KNN algorithm, only achieved an accuracy of approximately 42.88%. However, transitioning into a simple neural network brought the prediction accuracy up to 65.80%.

In the future, our team aims to attain higher accuracy by experimenting with different models and different feature extraction methods. In an article published in 2018 [7], the authors combined MFCC feature extraction along with Principal Component Analysis (PCA) to improve their speech recognition system. The addition of PCA reduced the number of MFCCs used by the model, which consequently increased the accuracy of their system from 86.43% to 89.29% [7]. We believe a similar approach, using PCA, can be used to improve the accuracy of our music genre classifier.

REFERENCES

[1] G. McDonald, Every Noise at Once. [Online]. Available: https://everynoise.com/everynoise1d.cgi?scope=all. [Accessed: 24-Mar-2021]

[2] Cataltepe, Zehra, Yusuf Yaslan, and Abdullah Sonmez. "Music genre classification using MIDI and audio features." EURASIP Journal on Advances in Signal Processing, 2007.

[3] Ş. Umut. "Automatic music genre classification using bass lines." 2010 20th International Conference on Pattern Recognition. IEEE, 2010.

[4] C. C. Coelho, B. Tilden, R. Rahman, W. Hasick. Music Genre Classification. https://wai-music-genre-classification.herokuapp.com/.

[5] Spectrogram. (2021, January 19). Retrieved October 16, 2020, from https://en.wikipedia.org/wiki/Spectrogram#:~:text=A%20spectrogram%20is%20a%20visual,as%20it%20varies%20with%20time.&text=Spectrograms%20of%20audio%20can%20be,the%20various%20calls%20of%20animals.

[6] Doshi, S. (2019, April 04). Extract features of music. Retrieved October 16, 2020, from https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d

[7] Winursito, Anggun, Risanuri Hidayat, and Agus Bejo. "Improvement of MFCC feature extraction accuracy using PCA in Indonesian speech recognition." 2018 International Conference on Information and Communications Technology (ICOIACT). IEEE, 2018.

3D Human Body Shape Generation

Ethan Bonnardeaux[1], Spencer Hill[2], Daniel Stewart[3], Noah Cabral[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: 18ejb@queensu.ca
2 e-mail: 18sjch@queensu.ca
3 e-mail: daniel.stewart@queensu.ca
4 e-mail: 15nsc4@queensu.ca

Abstract: *Generating point cloud models has become an increasingly popular practice within the machine learning community. Human shape data is the key to producing advancements within medical imaging, virtual reality, gaming, and animation fields. Learning object structure in 3-dimensional space presents many challenges in which deep learning networks have become iteratively capable of resolving. In this paper, we utilize a proven generative modeling technique to learn the approximate representation of human body shapes on point cloud data from the Semantic Body Models Dataset. By leveraging TreeGAN, a tree-based graph convolution generator network, our model is capable of learning the different segments of the human body in an unsupervised fashion. This approach combines the classic Generative Adversarial Framework with a nuanced generator that boosts its feature representation by sequentially accessing historical prediction states. Due to the consistent internal nature of human body shape data, we only sample points from the surface of the body, similarly restricting the model's learned representations.*

1. INTRODUCTION

1.1 Motivation

Recently, neural networks involving 3D data have attracted significant research interest. Since the introduction of Point Net in 2016, 3D point clouds have emerged as the most computationally efficient method of interpreting 3D data [1]. While most work has focused on object segmentation, classification, and object detection, in 2019 a novel architecture (TreeGAN) was proposed for 3D point cloud generation [2]. Leveraging the Generative Adversarial Network (GAN) [3] framework and tree-based graph convolution networks (GCNs), TreeGAN achieved seminal results on the ShapeNet40 dataset [2]. However, little to no work has been done to expand this object generation to more impactful datasets.

The immediate application of human body generation is to computer vision and medical imaging fields. The interpretation and generation of the human figure is an essential computer vision task that has received little attention. Furthermore, medical privacy restrictions make novel human body generation beneficial to training medical students and artificial intelligence systems on this data.

1.2 Related Works

1.21 Point Clouds, Neural Networks, and GANs

Most researchers transform point cloud data into 3D voxel grids or collections of images before running the data through deep learning pipelines. Charles et al. [1] proposed PointNet, a novel neural network that directly consumes point cloud data, which well respects the permutation invariance of point clouds. PointNet can be trained to perform 3D shape classification, shape part segmentation and scene semantic parsing tasks. Since the invention of PointNet, point cloud data have been used not just in classification networks but also in generative tasks. For example, Achlioptas et al. [4] proposed a GAN for the generation of 3D points clouds called r-GAN. The generator for r-GAN is based on fully connected

layers, leading to r-GAN having difficulty in generating realistic shapes with diversity.

1.22 Improved Training of Wasserstein GANs

A common issue in training GANs is the stability of training. Arjovsky et al. [5] introduced Wasserstein GAN, which uses an efficient approximation of the earth mover's distance function to optimize the discriminator and generator in GAN training. WGAN improved training stability and provided a meaningful loss metric that correlated with the generator's sample quality. However, WGAN still suffered from poor sample generation or a failure to converge, and it has been found that this is due to the weight clipping used to enforce a 1-Lipschitz continuous constraint on the critic. Gulrajani et al. [6] introduced gradient penalty, an alternative to weight clipping. It penalized the norm of the critic gradient with respect to the critics input, improving the sample quality and ability to converge for WGANs.

1.23 Graph Convolutional Networks

Over the past few years, many works have focused on using deep neural networks for graph problems. Defferrard et al. [7] proposed fast-learning convolutional filters for graph-based applications, significantly accelerating one of the main computational bottlenecks in graph convolution problems with large datasets. Kipf and Welling [8] introduced scalable GCNs, where convolution filters use only the information from neighboring vertices instead of from the entire graph. All the GCNs mentioned prior are designed for classification problems, meaning that the connectivity of nodes in the graph were known beforehand. This issue will present challenges for the generation of 3D point clouds, where the connectivity is not known.

1.24 GCNs and GANs for 3D Point Clouds Generation

A number of works have tackled the issue of connectivity. Valsesia et al. [9] dynamically generated adjacency matrices using the feature vectors from each vertex at each layer of graph convolutions during training. Unfortunately, computing this matrix at a single layer incurs a quadratic computational complexity on the number of vertices. This approach is not effective for multi-layer and multi-batch networks. Dong et al. [2] proposed TreeGAN, which, like the other work, requires no prior knowledge regarding connectivity. TreeGAN, however, is much more computationally efficient as it avoids constructing adjacency matrices. It uses a tree-based graph structure, and it exploits this structure by using ancestor information to propagate features over the graph. The tree-based graph structure also has the benefit of allowing the network to generate point clouds for different semantic parts of a model without prior knowledge.

1.3 Problem Definition

In this paper, we introduce point cloud generation of human body shape representations from randomized latent vectors. We explore the semantic parametric reshaping of human body models dataset [10] (a derivative of the Caesar dataset) to train our model. Historically, point cloud generation has been explored solely on the ShapeNet40 dataset. This dataset contains 40 different object classes and enables the generator models to produce a wide range of outputs. Currently, the TreeGAN paper has achieved state-of-the-art results on this dataset. We aim to train a generator on a single object class with a higher point-cloud resolution (3072 points) to produce increasingly granular results.

2. METHODOLOGY

2.1 Dataset

The dataset used to train and evaluate our model is composed of 3048 scanned body models. More precisely, there are 1531 male models and 1517 female models. To generate this dataset, we collected the mesh models from the publicly accessible dataset: Semantic Body Models Dataset [10]. The decision to develop our dataset from the Semantic Body Models Dataset is driven by the fact that it is open-sourced and completely available to the public and research communities. Further, the meshes in this dataset are pose-invariant, thus, allowing for more efficient learning of the true biological features of the human form rather than differences in pose.

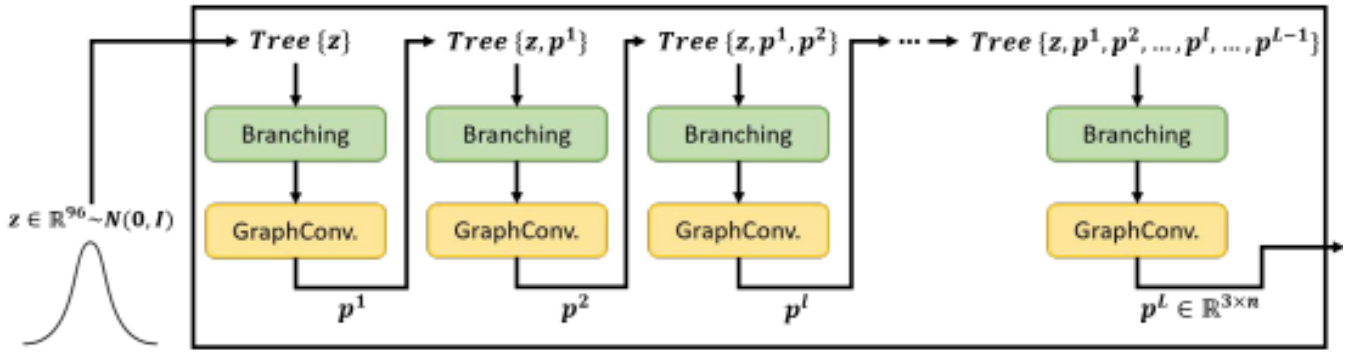


Figure 1 TreeGan Generator Architecture [2]

Mesh objects are highly memory-intensive due to the nature of their vertex-facet construction. Thus, point clouds from scanned meshes were built using an even surface sampling method to construct the point clouds from 3072 evenly spaced points on the surface of the mesh. This method of surface sampling was done for two reasons. The data points within the volume of the mesh do not significantly contribute to the learned features of the human form under the assumption that models are not hollow; Surface sampling maximizes the resolution of features using a point cloud representation while also minimizing memory and compute costs.

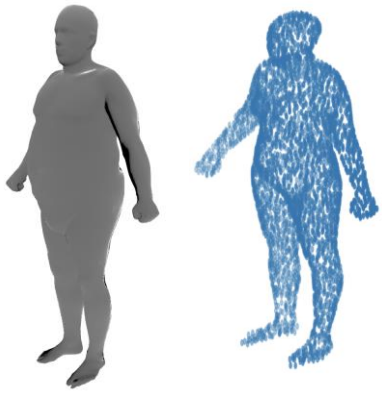


Figure 2 Data example from the Semantic Parametric Reshaping of Human Body Models dataset and a point cloud sample used in model training.

2.2 Model Architecture

Our model is built on the GAN framework, in which a generator and discriminator model train sequentially according to respective loss functions introduced in Wasserstein GAN [5].

$$L_{gen} = -E_{z\sim Z}\left[D\left(G(z)\right)\right], \quad (1)$$

$$L_{disc} = E_{z\sim Z}[D(G(z))] - E_{z\sim R}[[D(x)] + \lambda_{gp} E_{\hat{x}}\left[\left(\left|\left|\nabla_{\hat{x}} D(\hat{x})\right|\right|_2 - 1\right)^2\right]. \quad (2)$$

G and D denote the generator and discriminator networks, z is a latent vector created using a normal distribution, $\hat{x}$ are line segments between real and fake point clouds, $x' \sim G(z)$ and x represent real and fake point clouds respectively, and R is the real data distribution. We also apply a gradient penalty, λ_{gp}, to satisfy the 1-Lipshitz condition for GANs.

The generator leverages tree convolutions defined by

$$p^{l+1} = \sigma\left(F_K^l\left(p_i^l\right) + \sum_{q_j \in A\left(p_i^l\right)} U_i^l q_j + b^l\right), (3)$$

which is thoroughly described in (tree-GAN).

The generator takes as input a 96-dimensional latent vector, and through the convolution defined above, conventional convolutional neural network loop terms, and upsampling through defined branching, outputs a set of 3072 3D points. Figure 1 shows the generator built with tree-GCN layers.

As standard in GAN training, the Adam optimizer was used with the custom loss functions shown in (1) and (2).

3. RESULTS AND DISCUSSION

GAN evaluation metrics are an ongoing discussion within the research community as quantitative evaluation methods are continuously being introduced to measure crucial elements of a Generator's performance. Given the nature of this project, the appropriate evaluation metrics are Jensen-Shannon-Divergence (JSD), Coverage (COV-ED, COV-MMD), and Minimum Matching Distance (MMD) [11]. These metrics require a MMD comparison between the

reference data and the generator's closest representation per reference example. Our model is currently on ~epoch 500 within the training process and produces outputs as seen in figures 3, 4. Per qualitative examination, the generated results are not yet at comparable to the training data and minimum matching distance would provide inaccurate pairings between the generated and reference examples, resulting in a misleading evaluation of the model's efficacy.

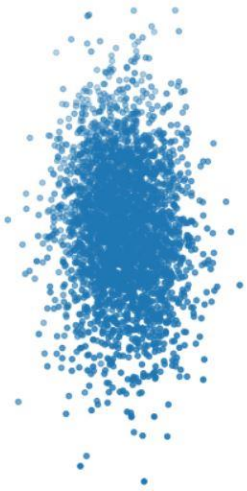


Figure 3 Generated Human Body Shape rotated 45 degrees along Z-axis.

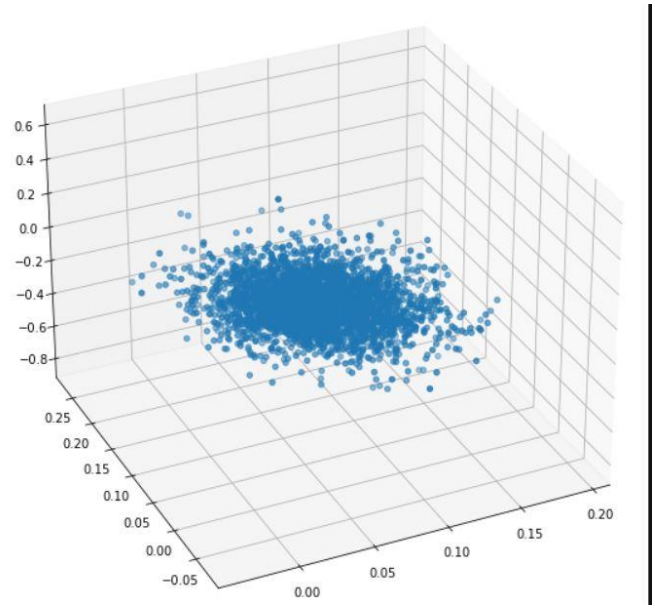


Figure 4 Alternate Generated Human Shape (Side View)

Upon observation, the model has clearly learned the basic features of a human body. Specifically, it has begun representing the chest, arms, legs, and head. As the training progresses, the amount of noise in the generated examples is expected to significantly decrease. The TreeGAN architecture was designed for non-hollow 3D point-cloud data. Due to the hollow composition of human shapes within our dataset, the generator has had difficulty minimizing its loss on the cylindrical-like components of the bodies.

4. CONCLUSIONS AND FUTURE WORK

In this project, we trained a TreeGAN model to introduce the generation of human body shapes to the machine learning community. We discovered a drawback when applying this architecture to hollow shapes. Future work on this problem should involve alterations within the TreeGAN architecture to effectively handle hollow data. Interpolation would also be an interesting area of exploration for a final model to permit controllable generation.

REFERENCES (IEEE format)

[1] Charles R. Qi et al. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.

[2] Dong Wook Shu et al. (2019). 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions

[3] Ian J. Goodfellow et al. (2014). Generative Adversarial Networks

[4] Panos Achlioptas et al. (2018). Learning Representations and Generative Models for 3D Point Clouds

[5] Martin Arjovsky et al. (2017). Wasserstein GAN

[6] Ishaan Gulrajani et al. (2017). Improved Training of Wasserstein GAN

[7] Michaël Defferrard et al. (2017). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

[8] Thomas N. Kipf et al. (2017). Semi-Supervised Classification with Graph Convolutional Networks

[9] Valsesia, Diego & Fracastoro et al. (2019). Learning Localized Generative Models for 3D Point Clouds via Graph Convolution.

[10] Yipin Yang et al. (2014) . *Semantic Parametric Reshaping of Human Body Models*. In 3DV Workshop on Dynamic Shape Measurement and Analysis.

[11] Panos Achlioptas et al. (2018). Learning Representations and Generative Models for 3D Point Clouds

Accessibility Bot

Michael Olson[1], Kevin Quijalvo[2], Kiera Lowman[3], Madeline Mackie[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada.

1 e-mail: michael.olson@queensu.ca
2 e-mail: 18mkq@queensu.ca
3 e-mail: 18knl2@queensu.ca
4 e-mail: 19mcm6@queensu.ca

Abstract: *This paper presents a potential solution for the lack of publicly available and reliable information on the accessibility of specific public buildings. We present a method to collect 3D models of an indoor space using a Turtlebot 3 and a IntelRealsense depth camera, and a method for accessibility feature extraction in the form of a convolution neural network trained on a dataset of indoor features related to accessibility.*

1. INTRODUCTION

1.1 Motivation

People living with physical disabilities, particularly those with motorized wheelchairs can have trouble accessing public spaces [1]. Currently the sources of information on the accessibility of specific public spaces are scattered and unreliable, and whether a public space is maneuverable for an individual is determined largely based on previous experience. Currently there is no publicly available database of 3D models of building interiors for the purpose of accessibility. Open source tools to map public spaces and present them with specific features highlighted exist and are widely available, but have not yet been applied to this specific purpose.

We hope to address the need for data related to the accessibility of public spaces by using modern mapping and feature analysis techniques.

1.2 Related Works

Several techniques have been developed for mapping interior spaces. Exploration of an Indoor- Environment by an Autonomous Mobile Robot was completed at the University of Kaiserslautern by Thomas Edlinger and Edwald Von Puttkamer [2]. This project chose to use a "Bubble Method". This method worked, through a virtual bubble forming with borders, which were determined by the scanner, forming around the robot. The robot was then programmed to explore the bubble causing the virtual bubble to deform until there were no more virtual borders. The Robot continuously searching for virtual borders ensured that new data of new locations was collected. While the robot was exploring data was also collected about the location of the real boarders. The results that were obtained from the project at the University of Kaiserslautern were that this method works in most indoor environments, as well as can be executed in real time, therefore is suitable to be installed on autonomous mobile robots.

There are many popular and publicly available solutions for classifying images. One of these systems, Residential Energy Service Network (ResNet) is a common deep learning method that can be used for image classification[4]. The development ResNet addressed the issue of the vanishing gradient problem, which is an issue that unrolls the network each input time step resulting in a very deep network that requires weight updates [3]. This issue was solved through the creation of residual blocks which allows connections to be skipped and the residual to be passed on to following layers. Overall allowing the neural networks to be more dynamic [4].

1.3 Problem Definition

There is currently no way for the approximately 300,000 Canadians aged fifteen and older who use a wheelchair to easily determine whether they will be able to access

small establishments such as restaurants and shops [5]. Their only options are to travel to the building to investigate or to depend on the potentially unreliable testimonies of business owners, online reviewers, and friends and family. We aim to develop a technology that will allow wheelchair users and other individuals with physical disabilities to have access to all of the information required to determine if a space is accessible. This includes specifications such as the width of doorways, the height of steps, and the location of handrails.

The objective of this project is to program a robot to autonomously explore and map an indoor space and identify features that promote or hinder accessibility. In order to successfully navigate the space, the robot must use a simultaneous location and mapping (SLAM) algorithm. The camera data would be sent from the Waffle Pi to a computer to create a 3D map and identify important aspects of the space. The solution must integrate SLAM, 3D mapping, and object detection to effectively aid people living with physical disabilities in their daily activities.

2. METHODOLOGY

2.1 Hardware Integration

Proper implementation of our idea meant that we needed to be able to control the robot remotely, and also to remotely receive image data from the camera attached to the robot.

The first step was configuring the Robot Operating System (ROS) on the robot. ROS was essential for us to interface with the robot's hardware, including the cameras, the wheels, etc. ROS uses a "topic" system that allows us to obtain information about the hardware and sensors (e.g. battery life, velocity, location, etc.). By "subscribing" to these topics, we can obtain the information. Through SSH, we remotely accessed the robot, and then downloaded and installed pre-built TurtleBot3 software directly onto it. The same software packages were also installed on another computer that was used to remotely control the robot. The pre-built TurtleBot3 packages already included a script that allowed us to control the robot with a keyboard and mouse.

Another crucial step was configuring the Intel RealSense camera, and setting up the robot to send its data to the laptop remotely. Since ROS was already being used to interface with other hardware on the robot, we naturally had to install the appropriate ROS wrapper for the Intel RealSense camera. The wrapper allows us to send the camera's data directly to the laptop using ROS's topic system. By subscribing to the appropriate camera topics using rviz (a visualization tool built into ROS), we were able to see the 3D models and images that the camera was generating in real-time.

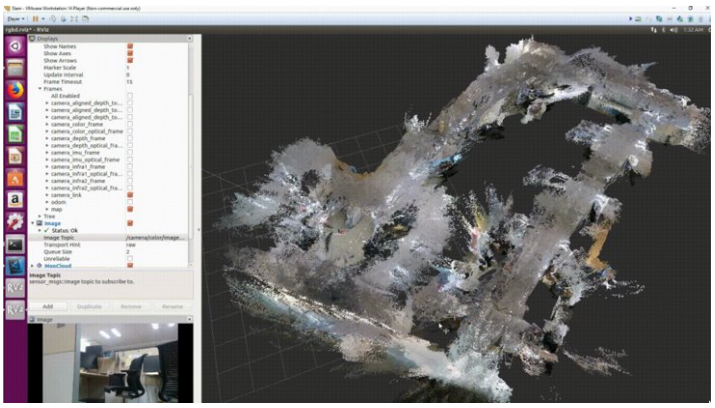


Figure 1: Example of a 3D model generated by the Intel RealSense camera in rviz.

2.2 SLAM

SLAM involves the robot being able to autonomously move and map the area without having a human control it.

By utilizing Intel's RealSense wrapper to obtain information from the camera, the data can be used with other open-source ROS localization packages to get the robot to move autonomously and map its surrounding area.

2.3 Dataset and Model Creation

It proved challenging to find an openly available dataset containing images of indoor building features that are important for accessibility. Consequently, a dataset was created using Google Images. We attempted to download images using the Javascript Console of Google Chrome, but the browser began blocking our attempts after a few successful trials. The images were instead obtained using a less efficient Google Chrome extension. The dataset contained 1500 images of stairs, handrails, and doors.

A machine vision model was built by retraining the Inception ResNet v2 model available through Keras, an open-source Python library. We chose to use this model because it uses a convolutional neural network (CNN), which provides very high accuracy for image classification, which has been demonstrated against the ImageNet dataset. The last Dense layer was changed to have an output dimensionality of 3, which corresponds to the number of classes that our model currently detects.

For validation, we randomly selected 20% of the images from our dataset.

Figure 2 above shows the last few layers of the model's structure:

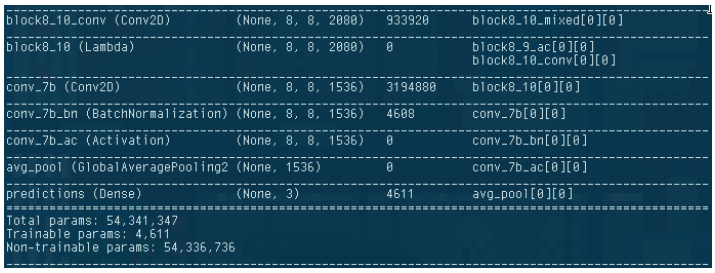


Figure 2: Last chunk of layers of the model structure

Note that the main difference between Inception ResNet v2 and our version of the model is the last layer.

3. RESULTS AND DISCUSSION

Using the modified version of Inception ResNet v2, we were able to achieve an accuracy of 75% and a loss of 5.8 against the validation images selected from our dataset.

After testing using images taken with a phone, and other images taken from Google Images, we found that the model struggled to classify the images reliably. It also struggled to classify images within the dataset we created. This was indicative of an underfitted model.

Due to the limited size of the dataset, and the naive approach to training, these results were about as expected. Our approach with changing the single layer had the advantage of being quick to implement, and it demonstrated potential. However, the accuracy of this current implementation is too low for practicality. Further model modification and fine-tuning, or even considering using or creating a different model architecture that caters towards our use case has the advantage of being more reliable, which is the ultimate goal.

Unfortunately, due to version mismatching with the open-source ROS libraries between the laptop and the robot, we were also not able to get SLAM functionality fully working. We were only able to move the robot manually using a keyboard and mouse, and had to map the robot's surrounding area under our control.

4. CONCLUSIONS AND FUTURE WORK

The goal of this project was to program a robot to autonomously map and explore a space and identify key features for accessibility. The retrained CNN recognized doors, stairs, and handrails in the validation set with an accuracy of 75%. The precision of the model could be improved by at least 20% through using a larger dataset and further refining the model. Further work on the hardware platform, and to the SLAM software is required. Additionally, further work is required in implementing a direct pipeline from the realsense camera to the CNN.

REFERENCES

[1] "Power wheelchair driving challenges in the community: a users' perspective," *Taylor & Francis*. [Online]. Available: https://doi.org/10.3109/17483107.2014.898159. [Accessed: 25-Mar-2021].

[2] T. Edlinger and E. von Puttkamer, "Exploration of an Indoor-Environment by an Autonomous Mobile Robot," 16-Sep-1996. [Online]. Available: https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/edlinger_exploration.pdf. [Accessed: 24-Mar-2021].

[3] J. Brownlee, "How to Fix the Vanishing Gradients Problem Using the ReLU," *Machine Learning Mastery*, 25-Aug-2020. [Online]. Available: https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/. [Accessed: 25-Mar-2021].

[4] R. Z. Zheng, "Beginners' Guide to Image Classification: VGG-19, Resnet 50 and InceptionResnetV2 with TensorFlow," *Medium*, 29-Apr-2020. [Online]. Available: https://cutt.ly/4xP1jmH. [Accessed: 25-Mar-2021].

[5] Statistics Canada, "Needs for mobility devices, home modifications and personal assistance among Canadians with disabilities," *Statistics Canada*, Aug. 16, 2017. [Online]. Available: https://www150.statcan.gc.ca/n1/pub/82-003-x/2017008/article/54852-eng.htm. [Accessed: Mar. 15, 2021].

Application of Neural Networks for Heart Disease Prediction

Ayushi Kardam[1], Emily Wang[1], Delaney Stevens[1], Jade Codinera[1], Jordan Ingram[1]

Western AI, Western University
Western University, London, ON N6A 3K7, Canada.

akardam@uwo.ca
dsteve47@uwo.ca
ewang56@uwo.ca
jcodiner@uwo.ca
jingram9@uwo.ca

Abstract: *Heart disease is a leading cause of death around the world, necessitating the development of software that can predict the likelihood of a person having heart disease in the future by analyzing medical data. This prevention method will ideally limit the number of heart disease patients. This application's purpose is to tackle this problem using three neural network models to predict a patient's likelihood of getting heart disease from patient attributes in the Cleveland database. Three neural network models were implemented: SVM, KNN, and RBF. The results of all three models were optimized to achieve accuracies between 64-95%. A long-term goal is to improve the models' accuracy to 90-95%.*

1 INTRODUCTION

1.1 Motivation

Heart disease is one of the leading causes of morbidity and mortality around the world, with numbers increasing every year; the mortality rate is around 31% annually. These statistics show that there is an increasing need for software to warn an individual the probability that he or she will develop heart disease in the future, so immediate action can be taken to live a healthier lifestyle. Complex algorithms and neural networks can be utilized through machine learning and artificial intelligence to enable self-learning from data without requiring human intervention.

1.2 Related Works

In a similar investigation into the use of neural networks for heart disease prediction, 6 ML classifiers were used to validate the Cleveland dataset. It was found that Chi-square and principal component analysis with RF had the highest accuracy overall. It was discussed that a major problem that frequents machine learning is the high dimensionality of the dataset and reducing this is key to higher accuracies, especially through the use of feature selections techniques [1].

1.3 Problem Definition

The goal of the Heart Disease Predictor is to apply different neural networks to improve upon the model's accuracy in the studies outlined in the Related Works and to create an interactive front-end for users to access this information.

2 METHODOLOGY

2.1 Data

First, start with presenting the data. The dataset that was analyzed is the Cleveland dataset obtained from the UCI ML repository. The dataset contains 14 medical parameters that were sorted through.

2.2 Proposed Solutions and Evaluation

2.2.1 SVM

The SVM algorithm is essentially a method of creating a hyperplane of best fit or decision boundary to sort data into separate categories. In the case of the heart disease predictor, a hyperplane of best fit divides patient data into either the prediction that the patient does or does not have heart disease [2].

Kernels are used to implement an SVM and each type of kernel has unique parameters that need to be tuned. Three kernels were used and optimal values for each kernel's parameters were found.

To find the optimal value of each parameter, the team created a program to loop through the SVM algorithm and output the accuracy for different parameter values.

2.2.2 KNN

KNN is a supervised machine learning algorithm that depends on input data to produce an output based on new unlearned data. This algorithm assumes that similar data will exist in a closer proximity. The calculation of similarity comes from calculating the distance between two points. The distance between the points as well as an index is then appended to an ordered collection. It is then sorted by ascending order of distances collected where the K values of the first few data points are selected. These K values in classification will return the mode [3].

Choosing the correct and most appropriate K value is crucial to improving the accuracy. The K value is modified until it reaches an optimized accuracy. Eventually, there will be more noticeable errors in the results, meaning that the K value has become too large. Conversely, reducing the K value to 1 reduces stability as the algorithm is observing a singular nearest neighbor. For this classification problem, K is typically an odd number to act as a tiebreaker [3].

2.2.3 RBF

The RBF Neural Network is a three-layer feedforward neural network using clusters with a smooth gradient. The first layer is an input layer, the second applies the RBF, and the third is the output layer which is determined by applying a weight (found using least squares linear regression) to layer two. The classification of an unknown point is determined by first finding the RBF vectors of the point with respect to their centroids (these centroids are found by performing k-means). The next step is to then apply a weight to each of these vectors and determine the maximum value. Finally, the index of this maximum value is then returned as the class that the unknown point belongs to.

The model was optimized by refining the hyperparameters beta, the speed of decay of the gradient, and k, the number of clusters, for maximum accuracy. The two equations considered for beta are shown below [4]:

$$\beta = \frac{1}{std^2} \quad (1) \qquad \beta = \frac{\sqrt{2 * k}}{Dmax} \quad (2)$$

Different values of k ranging from 0 to 250 clusters were tested for each beta equation.

3 RESULTS AND DISCUSSION

The highest accuracy of each model is shown in Table 1 below.

Table 1 - Summary of Model Accuracy

Model	***Maximum Accuracy***
SVM	88.5%
KNN	87%
RBF	89%

3.1 SVM

Many trials were completed by changing a parameter for each kernel. The accuracy changes because it affects the shape of the hyperplane. Figures 1 - 3 show the results of using a program to test many parameter values for each kernel. Table 2 summarizes the highest accuracies for varying the parameters of each kernel.

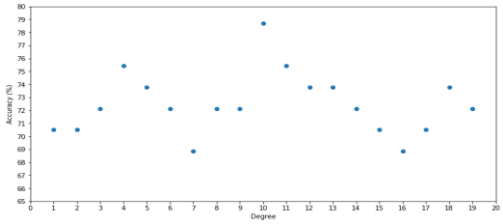


Figure 1 - SVM Polynomial Kernel

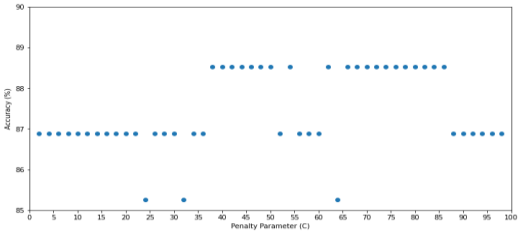


Figure 2 - SVM Linear Kernel

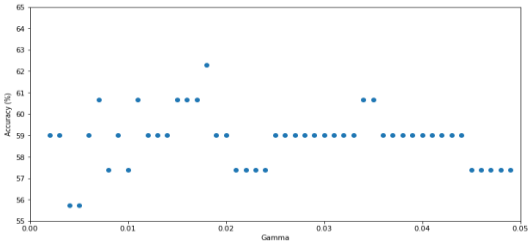


Figure 3 - SVM RBF Kernel

Table 2 - SVM Best Parameters

Kernel	***Parameter***	***Accuracy***
Polynomial	Degree = 9	0.7377
Linear	C = 75	0.8852459
RBF	Gamma = 1	0.590

The linear kernel with a C value from 60-80 resulted in the most accurate predictions. This range is due to the limited data as no data points lie within this range. A linear kernel with a C value of 75 was used for the Anvil application.

3.2 KNN

KNN is 64-95% accurate, showing the best result with K=13 at 87% accuracy. The team experimented with

all the odd values from 1-19 to see the accuracy at a large range of K values. As expected, there was some variability of the accuracy with each K value, so an approximate average accuracy value was recorded and summarized in Table 3 below. Each K value was re-run 10 times, and the high and low values were recorded for each [4].

Table 3 - KNN RBF Kernel

k	Avg. Accuracy Test Set (%)	Low Range Value (%)	High Range Value (%)
k=1	73	64	82
k=3	82	74	89
k=5	82	77	85
k=7	83	77	89
k=9	83	75	89
k=11	83	69	89
k=13	87	72	95
k=15	85	77	90
k=17	83	74	92
k=19	80	75	85

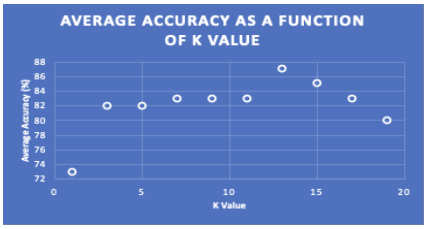


Figure 4 - KNN RBF Kernel

There are several advantages to this algorithm: it is easy to implement and simple since the user only needs to tune parameters and does not need to build the model from scratch. Although in this project it was used for the purpose of classification, it is super versatile and can be used for regression and search as well. The main disadvantage is that it should not be the chosen algorithm when predictions are required to be made rapidly.

3.3 RBF

The model accuracies for the different parameters are shown in Figure 5 and 6 below. A summary of the maximum accuracies from each beta equation are shown in Table 4. Based on this evaluation, the optimal beta was determined to be Eq. 2 with a k of 35 clusters which was implemented in the final model.

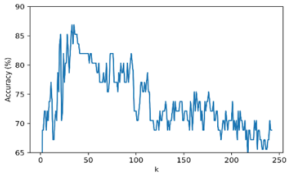


Figure 5 – Accuracies for Eq. (1)

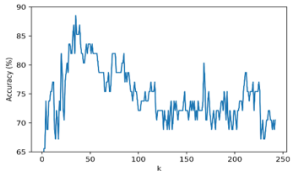


Figure 6 - Accuracies for Eq. (2)

Table 4 - Maximum Accuracy for Each Beta

Beta	k	Maximum Accuracy
Eq. (1)	33	86.89%
Eq. (2)	35	89.02%

Some advantages of RBF neural networks include higher accuracy and better efficiency. When dealing with noisy input data, RBF networks outperform conventional neural networks in terms of robustness and tolerance. Disadvantages include sensitivity to dimensionality and the possibility of not achieving the best performance due to a local minimum problem.

4. CONCLUSIONS AND FUTURE WORK

In conclusion, with the three models: SVM, RBF, and KNN, the highest accuracy in each model was optimized at 88.5%, 89%, and 87%, respectively. Since all the results were similar, and KNN fluctuated with a degree of error, the three methods were all presented on the website for the discretion of the user.

In the future, the group would like to optimize to accuracies in the range of 90-100% and develop an app that would allow medical professionals to use to confirm their heart disease diagnostics. One major challenge that the group faced was the large variability in the results; a way to improve this would be to obtain much more data, beyond the UCI heart disease dataset.

5. REFERENCES

[1] A. K. Gárate-Escamila, A. H. E. Hassani, and E. Andrès, “Classification models for heart disease prediction using feature selection and PCA,” *Informatics in Medicine Unlocked*, 27-Apr-2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352914820300125. [Accessed: 25-Mar-2021].

[2] U. Malik, "Implementing SVM and Kernel SVM with Python's Scikit-Learn", *Stack Abuse*. [Online]. Available: https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/. [Accessed: 05- Jan- 2021].

[3] O. Harrison, “Machine Learning Basics with the K-Nearest Neighbors Algorithm,” Medium, Jul. 14, 2019. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. [Accessed: 12- Feb- 2021].

[4] T. Ahadi, "Most Effective Way To Implement Radial Basis Function Neural Network for Classification Problem", *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/most-effective-way-to-implement-radial-basis-function-neural-network-for-classification-problem-33c467803319. [Accessed: 09- Jan- 2021].

Art Intelligence: Unpaired Image Translation using a CycleGAN

Danial Khan[1], Ashvin Ananthan[2], Jasdeep Singh[3], Abad Ullah[4], Hanna Dodd[5]

Western AI
Western University, London, Ontario N6A 3K7, Canada.

1 e-mail: Danial.Khan6312@gmail.com
2 e-mail: Aananth4@uwo.ca
3 e-mail: Jsingh.hba2022@ivey.ca
4 e-mail: Aawan26@uwo.ca
5 e-mail: hrdodd@gmail.com

Abstract: *Creativity and artistic expression have always set humans apart in terms of what makes us intelligent. As machines become more capable and intelligent, they can phase out tasks of increasing complexity. The goal of this work is to demonstrate a machine learning model's ability to recreate creative tasks, using the Cycle Generative Adversarial Network (GAN) architecture to create artwork out of photographs. Approaching image translation with a CycleGAN allows for the use of unpaired data while training. Utilizing the Adam optimizer with custom hyper-parameters, and the least squares loss function, this work is able to generate artwork to a similar degree as a human artist.*

1. INTRODUCTION

1.1 Motivation

Generative adversarial networks can be implemented into a wide variety of applications including, but not limited to image-to-image translation and generating realistic fakes of human faces [1]. The utilization of a cycle-consistent GAN allows for the use of unpaired image datasets. This unique feature of cyclic GANs eliminates the need to develop large datasets that contain paired images, which can be time exhaustive [2]. Through this deep learning network, one can transform an image by altering its stylistic elements without any prior artistic experience. Generative adversarial networks have a long list of real-world applications. This includes the generation of image datasets, realistic images, cartoon characters, and various aspects of photograph enhancement [3].

1.2 Related Works

The 2017 paper by J.Y Zhu et. al focuses on unpaired image translation using cycle-consistent generative adversarial networks. They engineered an approach that allowed one to translate a source image into a target image without the need for paired examples [2]. The foundation of the model depends on the adversarial loss and cycle consistency loss. It makes use of two generators and two discriminators that are consistently at battle with one another which brews the adversarial relationship. As the generator generates images for the discriminator from an input dataset, the same image is then put through a secondary generator that attempts to revert the image back to the original. The cycle consistency loss is an important aspect of the overall accuracy of the GAN. Their model handled the change in colour and texture very well, but geometric changes seem to be a challenge [2].

1.3 Problem Definition

The goal of this work is to create a cyclic GAN framework that maps landscape and architectural photographs to impressionist paintings similar to the works of French painter, Paul Cezanne. Due to the adversarial nature of the framework, standard ML assessment metrics such a loss and accuracy are less applicable. This is because the goal of each generator is to increase the loss of its respective discriminator vice-versa. GANs are a set of machine learning frameworks in which two neural networks, a generative (or 'generator') and a discriminative (or

'discriminator') network, compete in terms of the distribution of data. The generator maps data from a dataset to a data distribution of interest, while the discriminator takes this mapped data and tries to determine whether it coincides with the true data distribution. The goal of the generator is to increase the error rate of the discriminator, this makes the framework adversarial and greatly improves model accuracy [4]. A cycle GAN can be used to demonstrate the ability of an ML model to recreate creative tasks such as to create paintings from architectural landscapes.

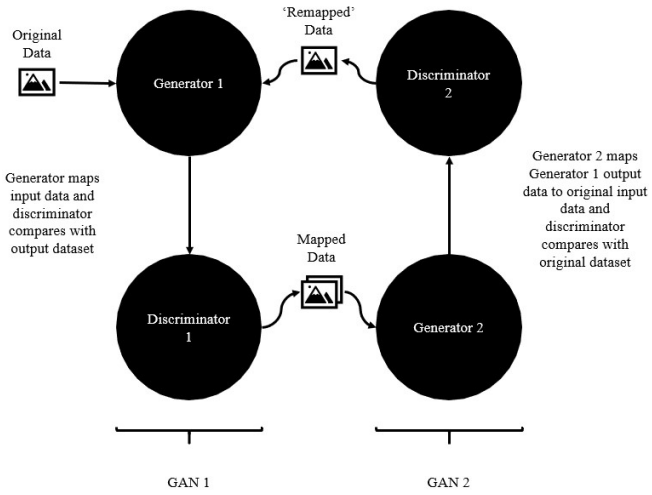


Figure 1: Components of a Cycle General Adversarial Network

2. METHODOLOGY

2.1 Data Collection

For image translation utilizing a CycleGAN two distinct datasets are required. The first data set contains data in the original format, in this case, real landscape photographs. The second data set contains our target format, Cezanne artwork. TensorFlow's built-in data set, Cezanne2Photo, was put into use providing training and test data split into both previously mentioned required sets.

Split	Examples
'testA'	58
'testB'	751
'trainA'	525
'trainB'	6,287

Figure 2: Dataset Contents

2.2 Generator Network

The generators are structured following the U-net structure described in the paper "U-Net: Convolutional Networks for Biomedical Image Segmentation" [5] where an image follows a series of down-sampling and up-sampling resulting in a remapped image. The generator down-samples the input image which consists of applying a convolutional layer and a max-pooling layer. Each down-sample causes the image dimensions to change while increasing the channel depth. Up-sampling utilizes convolution transpose layers that build the image back up to the original dimensions and channel depth.

2.3 Discriminator Network

The discriminators follow a fairly standard convolutional neural network structure. Each discriminator has a series of convolutions and pooling layers followed by a flatten layer so that the images can be run through a set of dense layers that output whether the image is real or fake.

2.4 Optimizer

The Adam optimizer is utilized for the training of the CycleGAN, this is to take advantage of the adaptive moment estimation. The model was originally trained using the following parameters for both the generators and discriminators as outlined by the original CycleGAN paper [2].

$$\alpha = 0.0002 \quad (1)$$
$$\beta = 0.5 \quad (2)$$

Equation 1, 2: Initial Optimizer Hyper-Parameters

Through varying trials of training, the initial learning rate, alpha, was kept constant to provide an equal starting point for each network. The beta value representing the first moment of gradient descent decay was lowered to 0.3 for the generator networks, this was after the observation that the discriminators constantly outperformed the generators.

2.5 Loss Function

Two loss functions were considered when developing both the generator and discriminator networks. The first was binary cross-entropy, to fit the categorical nature of the discriminator networks. The second was the least squares GAN loss as outlined by Xudong Mao, et al. in their 2016 paper titled "Least Squares Generative Adversarial Networks." [6].

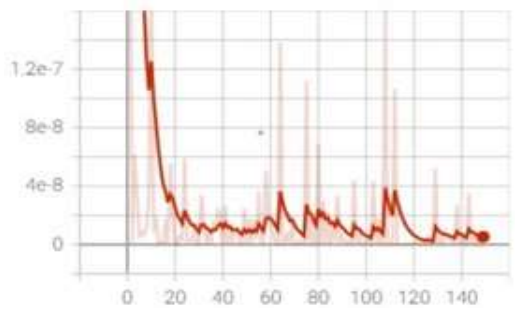


Figure 3: Binary Cross Entropy Discriminator Loss

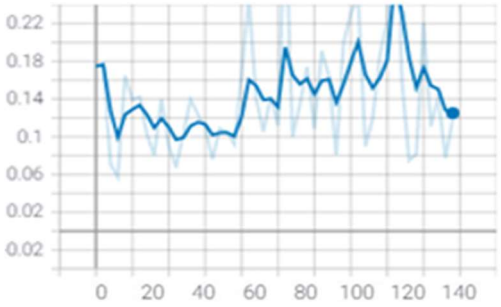


Figure 4: Least Squares Discriminator Loss

Figure 1 shows that with binary cross entropy the discriminator loss quickly approaches 0 and maintains a low value. Figure 2 shows that with least squares loss the discriminator loss continues to oscillate. Intuitively, binary cross entropy seems to be the better option but because of the desired adversarial nature of a CycleGAN, the oscillations in Figure 2 are more appropriate as it shows that the multiple networks are improving one another. These results determined that least squares loss would be used moving forward.

$$(D(x) - 1)^2 + (D(G(z)))^2 \quad (3)$$

Equation 3: Discriminator Loss

$$(D(G(z)) - 1)^2 \quad (4)$$

Equation 4: Generator Loss

3. RESULTS AND DISCUSSION

As mentioned previously, standard metrics such as accuracy and loss are not as applicable in the case of art generation as it is difficult to quantitatively determine whether an image passes as art. Figures 5, 6, and 7 display some of the generated images.




Figures 5,6,7: Generated Artwork

Major points of improvement for this model and a point of consideration for other CycleGAN applications would be the resolution of the images. With limited hardware resources obtaining high-resolution images became quite difficult. Aside from the resolution, the results are a strong indicator of AI's capability to automate more creative tasks such as art generation.

4. CONCLUSIONS

Overall, our CycleGAN was successful in translating landscape photographs into Cezanne paintings. We achieved a desirable outcome for the loss of both networks and can subjectively determine that the art produced was acceptable. Future steps for this project would be to attempt using datasets from other artists. This would create a visual comparison of how our model can mirror different art styles. By doing this, we would have a qualitative metric for how well the CycleGAN can produce art. We would be able to compare results in many different styles, and eventually may be able to create more unique and abstract art.

REFERENCES

[1] J. Brownlee, "A Gentle Introduction to Generative Adversarial Networks (GANs)", Machine Learning Mastery, July 2019.

[2] J.Y Zhu, T. Park, P. Isola, A.A Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks", ICCV, 2017.

[3] J. Brownlee, "18 Impressive Applications of Generative Adversarial Networks (GANs)", Machine Learning Mastery, July 2019.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," Communications of the ACM, vol. 63, no. 11, pp. 139–144, 2020

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," University of Freiburg, Freiburg, publication, 2015

[6] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, Stephen Paul Smolley; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017

Forest Ecosystem Analysis

Sarah Nassar[1], Adi Zingman[2], Henry Tsui[3], Mahad Rehan[4], Travis Cossarini[5]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada.

1 e-mail: 18sbn@queensu.ca
2 e-mail: 17az41l@queensu.ca
3 e-mail: henry.tsui@queensu.ca
4 e-mail: 19mmr1@queensu.ca
5 e-mail: travis.cossarini@queensu.ca

Abstract: *With the growing occurrence of natural disasters, such as wildfires, and dwindling natural resources, tracking the development of British Columbia's forests is of increasing importance. The purpose of this project is to identify and classify human disturbances, specifically tree cut blocks, with higher accuracy (greater than 90%) and consistency than human classification. Automatically identifying these geographic features provides a great benefit in terms of efficiency to technicians in the forestry and resource industries because manual classification is a time-intensive task. This furthers the progression in environmental technology and is a step towards using Artificial Intelligence to bring change in preserving and protecting the environment. Currently, an image segmentation algorithm using k-means clustering and a convolutional neural network model have been implemented to label areas where trees have been cut down from aerial images, with plans for an intuitive user interface.*

---

1. INTRODUCTION

1.1 Motivation

Identifying disturbances is a key process in forest ecosystem dynamics [1]. They strongly influence the structure, composition, and proper functioning of forest ecosystems [2]. They also determine the spatial and temporal patterns of forested landscapes [3]. Furthermore, disturbances are relevant factors in the management of ecosystems for functions, goods, and services [4]. These disturbances need to be found and classified to better understand the forest ecosystems. Normally these disturbances are manually classified by humans, but this solution is not optimal because it can be labour-and time-intensive. The purpose of the forest ecosystem analysis software is to be able to identify and classify tree cut blocks. The project aims to use object detection and image segmentation to create an algorithm that will classify these disturbances with higher accuracy and consistency than human classification.

1.2 Related Works

While the monitoring of forest resources is still in its infancy as an application of data science and machine learning, there have been applications of it in some S. states. In the peer-reviewed article, *Satellite Inventory of Minnesota Forest Resources*, the team was able to achieve a classification accuracy of 75% in classifying the amount of forested area across 6 forest classes [5].

studies
This was done through a mix of primary sampling unit (PSU) sampling and disturbance classification using two-phase, stratified sampling. While this application is similar to the one discussed in this paper, the execution does not include any traditional machine vision clustering techniques.

Figure 1: An aerial image (left), its labels (middle), and the corresponding cleaned cluster (right).

1.3 Problem Definition

The client uses aerial images taken from planes and satellites to conduct analyses of natural areas to guide prediction. They face the issue of having to manually analyze and identify human disturbances. This process is quite tedious and expensive. To solve this monotonous task, the objective of this project is to identify and classify tree cut blocks through the use of computer vision. Thus, the process of identifying the traits of a natural area would be automated and would yield a higher accuracy and consistency compared to human classification. Ideally the model must yield results that are considered accurate without the need for a human to verify the outcome.

2. METHODOLOGY

The client provided the team with aerial images in the MrSID file format and labels in the TIFF format. The aerial images were around 100 megabytes (MBs) each. The Python library used for reading the raster images is the Geospatial Data Abstraction Library (GDAL).

The first solution is an unsupervised one using k-means clustering in the Open Source Computer Vision (OpenCV) library. K-means clustering tries to group similar data points into clusters. The images were converted to grayscale for reduced memory consumption. The algorithm worked on a computer with at least 16 gigabytes (GBs) of memory. The required output cluster was cleaned using Gaussian filtering followed by Otsu's thresholding. *Figure 1* shows an input image, its labels, and the k-means clustering output.

The other solution is using a supervised method, namely a convolutional neural network (CNN). The model is U-NET with the VGG11 Encoder. The images were resized and split into tiles for reduced memory consumption. The algorithm worked on Google Colab's graphics processing unit (GPU) with less than 13 GBs of memory. 80 percent of the tiles were used for training and 20 percent were used for testing, and the tiles were shuffled to be randomly selected. *Figure 2* shows a sample tile with its labels and model output. The tiles would have to be merged and resized again once the algorithm is finished.

To evaluate the proposed solutions, a confusion matrix from the Python library, scikit-learn, was used. The output was compared to the client's labels to determine if overlapping pixels had the same label (i.e., black/positive or white/negative). *Figure 3* shows the confusion matrix framework for this project.

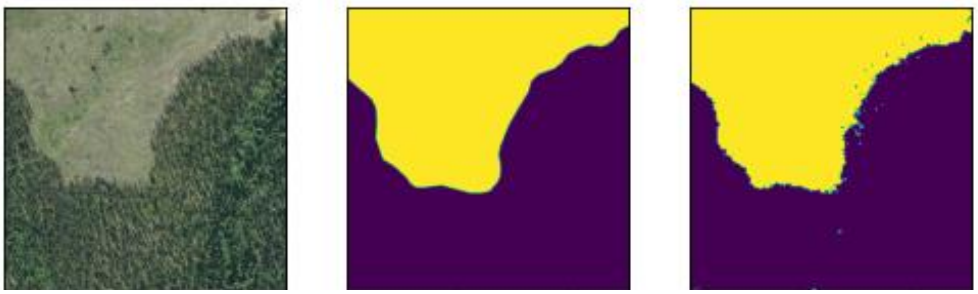


Figure 2: A sample tile (left), its labels (middle), and the predicted labels from the CNN model (right).

	Actual Black	Actual White
Predicted Black	True Positive	False Positive
Predicted White	False Negative	True Negative

Figure 3: The defined confusion matrix for this project.

3. RESULTS AND DISCUSSION

The accuracy, precision, recall, and F-score were calculated for the k-means clustering algorithm and the testing component of the CNN model using the values from the criteria matrices. The results are shown in *Table 1*.

Both solutions yielded high metrics, but the CNN performed better. For k-means clustering, the right cluster has to be manually identified from all the output clusters, the number of clusters and iterations required may vary, and there is no way to ensure consistent results because it is an unsupervised method. For the CNN model, there is training involved, which means that more consistent results are possible, and there is less human interaction so the process can be more automatic and integrated into a simpler user interface.

Model	***K-Means***	***CNN***
Accuracy	96%	98%
Precision	85%	84%
Recall	71%	96%
F-Score	77%	89%

Table 1: Results of the k-means clustering algorithm after de-noising and the CNN testing subset using the values from the confusion matrices.

4. CONCLUSIONS AND FUTURE WORK

The work the team has completed so far is excellent, though it will need to be formalized in order to be delivered to the client. The accuracy scores listed above prove the soundness of this application, though these scores will need to be validated across a wider set of images moving forward. The CNN system will be implemented due to increased accuracy and removal of the need for user input, though memory constraints are an important issue to consider.

Usability is also an important goal for the client, as users need not be experienced in programming or machine learning. Accordingly, the development of a basic user interface is also crucial for the success of the system.

REFERENCES

Acknowledgment: This work could not have been done without the assistance of a member of the analytics team at the Queen's University Centre for Advanced Computing (CAC).

[1] C.D. Oliver, B.C. Larson, Forest Stand Dynamics, Wiley, New York (1996), 520 pp

[2] J.F. Franklin, T.A. Spies, R.V. Pelt, A.B. Carey, D.A. Thornburgh, D.R. Berg, D.B. Lindenmayer, M.E. Harmon, W.S. Keeton, D.C. Shaw, K. Bible, J. Chen, Disturbances and structural development of natural forest ecosystems with silvicultural implications, using Douglas-fir forests as an example, For. Ecol. Manage., 155 (2002), pp. 399-423

[3] R.T.T. Forman, Land Mosaics: The Ecology of Landscapes and Regions, Cambridge University Press (1995), 656 pp

[4] K.J. Puettmann, K.D. Coates, C. Messier, A Critique of Silviculture. Managing for Complexity Island Press Washington, DC (2009), 206 pp

[5] M. Bauer et al., "Satellite Inventory of Minnesota Forest Resources," p. 12.

Financial Statement Analysis using Unsupervised and Supervised Learning

George Trieu[1], Jackson Kehoe[2], Alexia Tecsa[3], Raisa Sayed[4], Nicolas Wills[5]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: g.trieu@queensu.ca
2 e-mail: 17jpk3@queensu.ca
3 e-mail: alexia.tecsa@queensu.ca
4 e-mail: raisa.sayed@queensu.ca
5 e-mail: 17nvw@queensu.ca

Abstract: *With the growing number of self-investors in the financial markets today, there must be more tools to help investors make informed decisions. The goal of this project is to group similar companies together and perform basic peer analysis and comparison on these companies. This is achieved by using various clustering methods to group companies together, and finally, perform analysis on these clusters through supervised learning and Shapley values. The attributes of each company selected to perform clustering were important accounting ratios to determining a company's success - both financially and on the stock market. The final model uses affinity propagation clustering and produces thirteen final clusters.*

---

1. INTRODUCTION

1.1 Motivation

In 2020, more than 2.3 million Canadians opened investing accounts [1]. This trend was found all throughout North America, as the lead personal investing platform, Robinhood, alone saw 13 million new traders in the past year [2]. As a result of these trends, there has been a growing concern as to the responsibility of non-professional traders, and the lack of knowledge behind their investment decisions. This has resulted in many Canadians being placed in positions of high financial risk. As interest in stock investments continues to grow among the public, it is important to provide stock analysis tools to create more informed decisions, thus placing the public at less financial risk.

1.2 Related Works

There have been numerous attempts to build comparative financial analysis tools for equities in the stock market using machine learning techniques. In 2015, the Marbaselios College of Engineering developed a clustering and regression model for stock prediction [3]. The research developed demonstrated that partitioning-based clustering performed better than density-based clustering and hierarchical-based clustering. Another similar project was developed by the Intel Institute of Science [4]. In the project developed by Intel, hierarchical agglomerative and recursive k-means clustering was effectively used to predict the short-term stock price movements after the release of financial reports.

1.3 Problem Definition

The problem tackled in this project is to develop a method to compare and group different publicly traded companies. This was achieved by employing various clustering models to separate and classify companies; as well as by determining the key financial metrics in each grouping.

2. METHODOLOGY

This project was completed in three phases: data preparation, clustering technique experimentation, and supervised learning.

2.1 Data Collection and Preparation

A dataset from kaggle.com containing 200+ Indicators of US Stocks from 2014-2018 [5], was used for this project. This dataset was then prepared for use by eliminating blank "Not a Number" (NaN) value rows and narrowing down the number of attributes contained within the dataset. The accounting ratios (attributes) used were those that investors would most commonly use to assess the financial performance of any company as this would allow us to better analyze firms across industries. These would include earnings per share and the price-earnings ratio which most investors are typically concerned with. The data was also normalized prior to clustering.

2.2 Clustering Models

Once the dataset was cleansed, various clustering techniques were applied to it such as k-means, DBSCAN, spectral, agglomerative, Gaussian mixture, and affinity propagation. Each clustering technique was then compared to one another through the quality of the clusters generated. For example, DBSCAN yielded most of the companies forming in one singular cluster, which is not useful for the purposes of this project.

2.3 Supervised Learning

Once the most effective clustering technique was selected, the Random Forest supervised learning was trained using the cluster number as the target attribute, to learn more about the feature importances of the clusters. Shapley values were also used to further explain the significance behind each attribute in the model.

3. RESULTS AND DISCUSSION

3.1 Clustering Attributes

The attributes (accounts/ratios) of the dataset were truncated to only keep the ones that were useful. The final list of accounting ratios used for clustering is shown in Table 1.

Table 1: Accounting Ratios used for clustering the data.

Net Cash Flow/ Change in Cash	Average Payables	Average Receivables
Current Ratio	SG&A to Revenue	Days of Payables Outstanding
Days of Inventory Outstanding	EBIT per Revenue	Debt to Assets
Debt to Equity	Payout Ratio	Return on Equity
R&D to Revenue	PE Ratio	Dividend Yield

After cleaning the dataset and isolating for the above attributes, 3568 companies remained for use in the clustering process.

3.2 Clustering Methods

The clustering methods experimented with were k-means, DBSCAN, agglomerative, spectral, Gaussian mixture and affinity propagation. Of the following techniques only Gaussian mixture and affinity propagation yielded distinct clusters between the companies due to the large variation within the normalized data. The unsuccessful techniques yielded most of the companies forming in the same cluster or the marking of most data as noise.
The affinity propagation clustering method works by comparing the different data points within the data to each other using matrices. When two points attributes are similar enough, they form a criterion matrix for the newly formed cluster which other data points must satisfy to join this cluster. This method eliminates the need to specify the number of clusters and different numerical metrics.
The affinity propagation clustering formed 46 clusters and had a silhouette coefficient score of 0.367. Some of the clusters formed with limited data points which indicated noise within the data. These clusters were filtered out and resulted in 13 distinct clusters.

```
Out[16]: 2     1498
         1      931
         5      441
         6      206
         4      100
         8       97
         0       93
         7       52
         3       21
         10      16
         11      11
         9       11
         12      10
         Name: cluster, dtype: int64
```

Figure 1: Results from affinity propagation clustering. The cluster number is on the left and number of companies in each cluster is on the right.

3.3 Random Forest and Shapley Values Analysis

After the formation of the clusters, they were analyzed using a combination of random forest and Shapley values to identify the dominant features. The random forest gave a very high accuracy score of 94.74% in identifying which cluster each company belonged to. The most influential features in the formation of clusters were debt to assets, current ratio, and debt to equity.

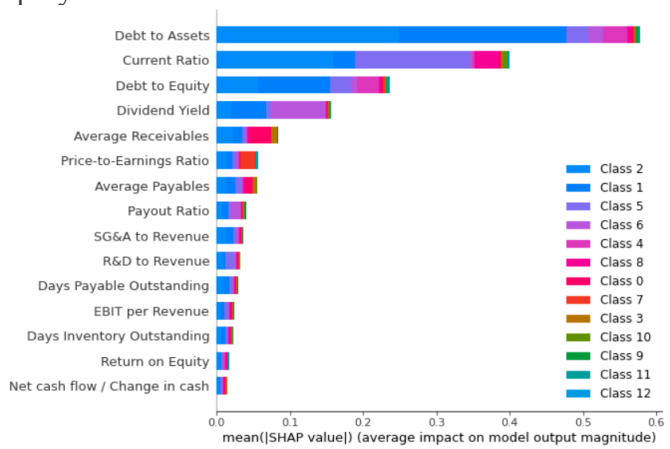


Figure 2: Contribution of features to the formation of different clusters.

Shapley values incorporate the model produced by the Random Forest algorithm to measure the contribution of a feature in each cluster individually. This is done using coalition game theory, by measuring the importance of each attribute to the predicted value [6]. An example can be seen in cluster 5, where most pharmaceutical and research companies were grouped.

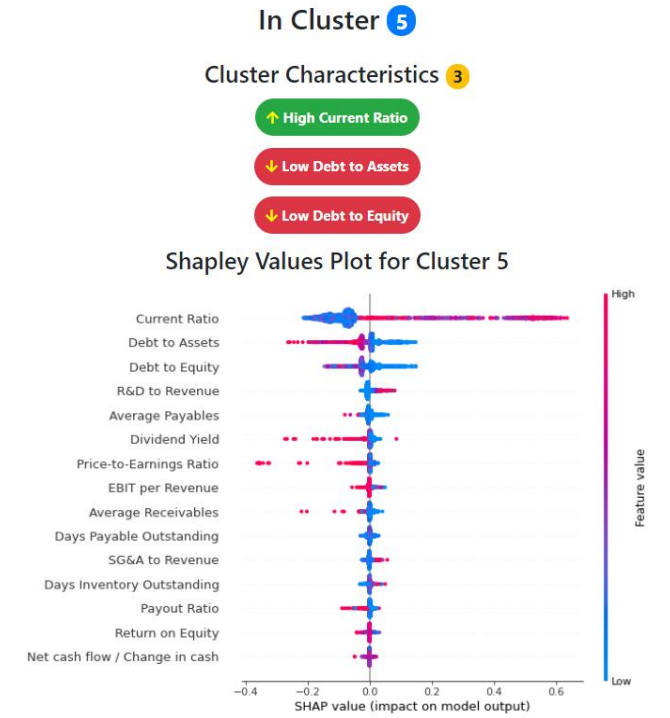


Figure 3: Sample results for cluster 5.

4. CONCLUSIONS AND FUTURE WORK

Without time constraints, further expansions to the project can be considered.

The first consideration would involve examining the companies within each cluster at a far more granular level to get a better understanding of their financial structures and the industries they operate in. This would allow the algorithm to further refine each industry cluster and identify closer similarities between companies within the same cluster.

The second application would be the analysis of the relative importance and impact of qualitative financial information on investment decisions. This analysis relies on non-quantifiable information such as management expertise, industry cycles, the strength of research and development, and labor relations [6]. Natural language processing can be applied to analyze textual data taken from management letters, financial statement notes, and other disclosed company announcements.

REFERENCES

[1] Global News, "Canadians opened 2.3 million DIY investing accounts in 2020. Should you?," 11 February 2021. [Online]. Available: https://globalnews.ca/news/7631776/diy-investing-canada-iiroc/.

[2] CNBC, "How Robinhood and Covid opened the floodgates for 13 million amateur stock traders," 7 October 2020. [Online]. Available: https://www.cnbc.com/2020/10/07/how-robinhood-and-covid-introduced-millions-to-the-stock-market.html.

[3] T. M. B.S. Bini, "Clustering and Regression Techniques for Stock Prediction," *Procedia Technology,* vol. 24, pp. 1248-1255, 2016.

[4] N. G. B. S. M.S. Babu, "Clustering Approach to Stock Market Prediction," *Int. J. Advanced Networking and Applications,* vol. 3, no. 4, pp. 1281-1291, 2012.

[5] N. Carbone, "200+ Financial Indicators of US stocks (2014-2018)," 2020. [Online]. Available: https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018.

[6] G. B. G. L. Huy Quan Vu, Data Mining Applications with R, 2014.

[7] T. Smith, "Qualitative Analysis," 7 March 2021. [Online]. Available: https://www.investopedia.com/terms/q/qualitativeanalysis.asp.

Forecasting Hospital Bed Occupancy

Aidan Turnbull[1], Jake Egan[2], Courtney Orcutt[3], Jacob Seiler[4], Mackenzie Sharp[5]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: aidan.turnbull@queensu.ca
2 e-mail: jake.egan@queensu.ca
3 e-mail: 18cro@queensu.ca
4 e-mail: jacob.seiler@queensu.ca
5 e-mail: 17mcs16@queensu.ca

***Abstract**: To combat rising health care expenses and wait times across Canada, our team in conjunction with Kingston Health Science Center developed a time series forecasting model to predict hospital bed occupancy. Using Facebook's Prophet library as a framework, the model was trained and tested with Kingston General Hospital occupancy data and turned into a web application for easy administrative use. Forecasting 365 days in advance the model displayed a mean absolute percentage error of 10.34 and a root mean square error of 11.31. Overall, the model effectively predicted bed occupancy and can easily be implemented by the Kingston Health Science Center to help prepare for surges as well as make their schedule more efficient.*

---

1. INTRODUCTION

1.1 *Motivation*

Canada has a hospital bed problem. The country ranks 34th in available hospital beds [1] and has a funding model that financially punishes hospitals for operating over capacity. These factors contribute to high patient wait times [2], overworked nurses [3], and underfunded hospitals. Many of these issues could be mitigated if hospital administrators had a better understanding of occupancy trends. Knowing how many patients to expect would allow the hospitals to move patients to less busy facilities, schedule an appropriate number of nurses, and save the hospitals thousands of dollars per bed [4]. Our goal is to create a time series forecasting model that can predict hospital bed occupancy accurately and easily be implemented by health care administrators.

1.2 *Related Works*

This program is a continuation of a QMIND project in 2020 with the same objective. That team used an autoregressive integrated moving average (ARIMA) model and was able to achieve extremely accurate results. Unfortunately, their model required being retrained daily which inhibited its ability for long term predictions.

1.3 *Problem Definition*

To maximize capability, hospitals need to know how many patients to accept each day. This information will allow them to schedule elective surgeries efficiently, know how many employees are needed, and prepare for expected surges. By expanding forecasting models to predict long term bed occupancy hospitals will have useful information to help optimize their operations.

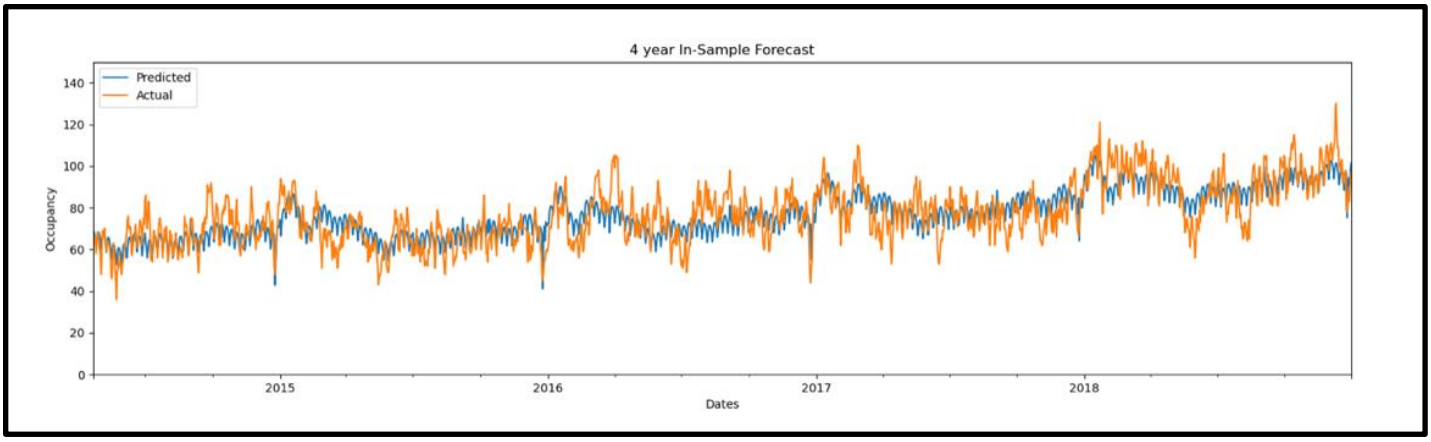


Figure 1. *Comparison of predicted (Blue) vs. actual (Orange) bed occupancy at Kingston General Hospital from April 21st, 2014 to March 31st, 2018.*

2. METHODOLOGY

The Kingston Health Science Center provided occupancy data from Kingston General Hospital between April 2014 and March 2019. Each observation included the date, admissions, discharges, day of the week, as well as if Queen's University was in session or on holiday.

This data set trained a time series forecasting model built on python with Facebook's Prophet library. The Prophet neural network was developed by Facebook to predict groups users would likely join based off their page likes and expressed interests. We chose Prophet because of its forecasting proficiency and it seamlessly integrating with our web application. The algorithm trained multiple times using different combinations of variables to find the most accurate forecasts.

Accuracy was calculated by comparing the predicated and actual occupancy over the course of 365 days.

Finally, a user-friendly web application was built that allows the model to be retrained on new data as well as forecast occupancy data for any specified time frame.

3. RESULTS AND DISCUSSION

Table 1. *Statistical results from 365-day forecast using Prophet model.*

	Mean Absolute Percentage Error	Root Mean Square Error
365 Day Forecast	10.34	11.31

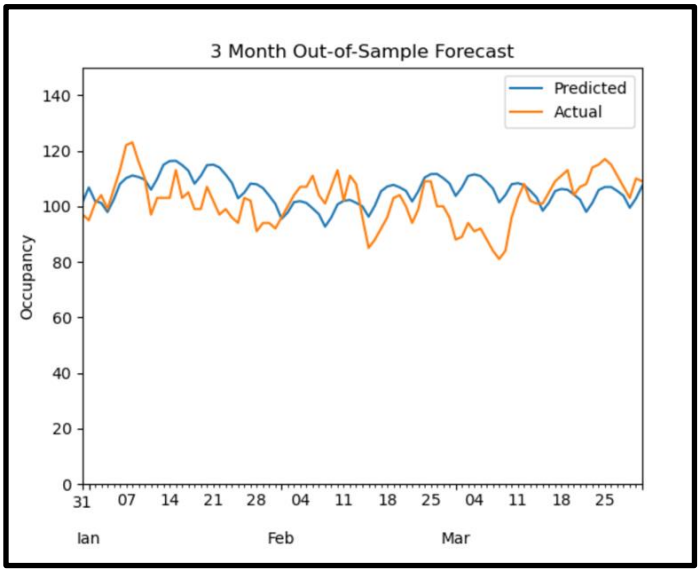


Figure 2. *Comparison of predicted (Blue) vs. actual (Orange) bed occupancy at Kingston General Hospital from January 1st, 2019 to March 31st, 2019.*

The 365-day forecast had a mean absolute percentage error of 10.34 and a root mean square error of 11.31 (Table 1).

The statistical results suggest the model can accurately forecast bed occupancy 365 days in advance. The four-year forecast (Figure 1) shows the model generally follows monthly trends including a sharp decrease during the December holidays and a gradual increase after Canada Day. Furthermore, the smaller three-month forecast (Figure 2) demonstrates the model takes weekly trends into account with Sundays usually being the least busy day. These figures also display limitations to the model, particularly that it always predicts a seemingly average number of patients. The four-year forecast shows an occupancy spike in April

2016 which the model failed to detect. This is partially due to the nature of hospitals with accidents occurring randomly causing a surge of patients, however, the model does not have access to what each bed is being used for which could give insight into why those spikes occurred and if they could have been predicted. Keeping these limitations in mind the model is not accurate enough to replace human judgement and automate the hospital's schedules, however, is effective enough to be a useful tool to augment administrator's expertise and should help optimize operations.

4. CONCLUSIONS AND FUTURE WORK

This project attempted to create a time series forecasting model that could accurately predict hospital bed occupancy. Using Facebook's Prophet library, the team built and trained a model that could forecast bed usage 365 days into the future with a mean absolute percentage error of 10.34. Although the model had difficulty anticipating large surges it was effective at predicting monthly and holiday trends. Ultimately, the model achieves its goal of effectively predicting hospital bed occupancy.

Future forecasting efforts should research occupancy of different departments within the hospital. Knowing which beds are being used could provide the model valuable information to better predict surges and generally improve the model's accuracy.

REFERENCES

[1] OECD, "Hospital Beds (Indicator)," *Organization for Economic Co-operation and Development*, Paris, France, 2019. [Online]. Available: https://data.oecd.org/healtheqt/hospital-beds.htm [Accessed: Mar. 24, 2021].

[2] A. Hildebrandt, "Hospital ER times reveal some 'disturbing' waits," *CBC*, Sep. 18, 2014. [Online]. Available: https://www.cbc.ca/news/health/hospital-er-times-reveal-some-disturbing-waits-1.2767867. [Accessed: Mar. 24, 2021].

[3] S. Fitzgibbon, "Work Stress Among Nurses in Ontario," *Queen's University Industrial Relations Centre*, vol. 2006-05, pp. 1-30, June 2006.

[4] S. Lunn, "Seniors in hospital beds costly for health system," *CBC*, Dec. 1, 2011. [Online]. Available: https://www.cbc.ca/news/health/seniors-in-hospital-beds-costly-for-health-system-1.1069802. [Accessed: Mar. 24, 2021].

Happy Recommender

Eric Yuyitung[1], Kevin Subagaran[2], James McCarron[3]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: eric.yuyitung@queensu.ca
2 e-mail: 17jmp5@queensu.ca
3 e-mail: 19kks1@queensu.ca

Abstract: *In our modern age, the extent of data collection is ever increasing. Being able to leverage this information has led to great improvements in overall user experience. An obvious example would be the "customers also bought" feature on Amazon. This is a recommender system and helps tailor the experience to the user, promoting engagement and convenience. Unfortunately, high quality recommender models are cumbersome to implement. To solve this, we have created a library called Happy Recommender which streamlines the implementation of recommender models. The project is built on Tensorflow and implements a state-of-the-art model in a matter of lines of code. This project serves as a high-quality foundation and from there will be developed further with client Vennify Inc., to improve performance, usability, and value, to help the project contribute to, and create impact within, the AI community.*

1. INTRODUCTION

1.1 Motivation

Looking at the modern digital world, many products and services leverage abundant data to tailor the user experience (UX). A 2013 article titled "The Importance of Personalization in E-Business Environments" outlines the advantage of data driven UX optimization and it covers successful examples such as, Facebook's "people you may know".

Under the hood, these optimizations are driven by recommender systems, but due to performance constraints in their typically web-based applications, these have formerly been simple user segmentation models and shallow pattern matching or rule-based techniques [1]. However, recent advances in deep learning have led to deep recommender systems, generating state-of-the-art results, more effectively incorporating contextual information [2].

Given the importance of these systems in modern UX, we wanted to implement a deep recommender system for a new social media app Kaku and were met with the complex task of implementation. So, the scope was altered to create a library to streamline the implementation of state-of-the-art recommendation systems.

1.2 Related Works

Upon initial research into methods and tools to use to develop this tool, Python and Tensorflow were selected as the foundation. Shortly into development, Google released the Tensorflow Recommenders library which initially seemed to encompass the project goals but was largely a convenience library, still requiring much of Tensorflow's boilerplate code to use effectively [3].

Further into development we came across a similar tool, Tensorrec. It accomplished a very similar project goal but was built on an old Tensorflow methodology (directly manipulating graphs) and unfortunately, has since also been deprecated [4]. Inspiration for the user's interaction with library was drawn from this project, in terms of what methods make sense to expose.

1.3 Problem Definition

The power of recommender systems is made apparent by their essentiality in modern UX and yet, the few existing tools, are too complicated for amateurs or lack streamlined implementation.

The team aimed to create an open-sourced library which facilitated usage of modern recommender models across a wide range of data while still being simple to use. We focused on implementing deep collaborative-filtering as it is a powerful and intuitive methodology which is applicable to a wide range of dataset tasks. This was then supported by the creation of high-quality documentation to further increase the accessibility of the library.

2. METHODOLOGY

2.1 Solution

It was decided the most robust implementation would be a wrapper on top of Tensorflow Recommenders. As a google supported project, it would be a reliable dependency and offers the most robust functionality despite its relative difficulty to use.

To simplify usage, Happy Recommender uses reasonable default parameters and object-oriented programming (OOP) to minimize the boilerplate code required from the user. When Happy Recommender is initialized with a dataset, all data processing and splitting, model definition and compiling is handled, and a compiled model is returned to the user. From there a user need only specify the quantity of training epochs, then the trained model is ready for evaluation or implementation. Should an advanced user want greater control over the model, many of these parameters are exposed as optional function inputs.

```
from happyrecommender.happy_matrix import HappyMatrix

# model is initialized and compiled on input dataset
h = HappyMatrix(path='../datasets/movielens/data.csv',
                user_id='userId',
                content_id='movieId',
                rating_id='rating')

# model is trained on train split of dataset
h.model.fit()

# model is evaluated on test data
results = h.eval()

#predicted rating is generated for user 232 on content 5
prediction = h.predict(user_id=232, content_id=5)
```

Figure 1: Example of code required for usage

2.2 Design Process

The design process began with a wide breadth of research into models and datasets to ensure that the selected model can gracefully handle as many datasets as possible. A two-tower deep embedding model was selected, (as seen in Figure 2) as it is applicable across many modern datasets and is a powerful state-of-the-art deep recommendation model [5]. From this model we built out the required infrastructure, namely data preprocessing, model training, model evaluation, and recommendation generation.

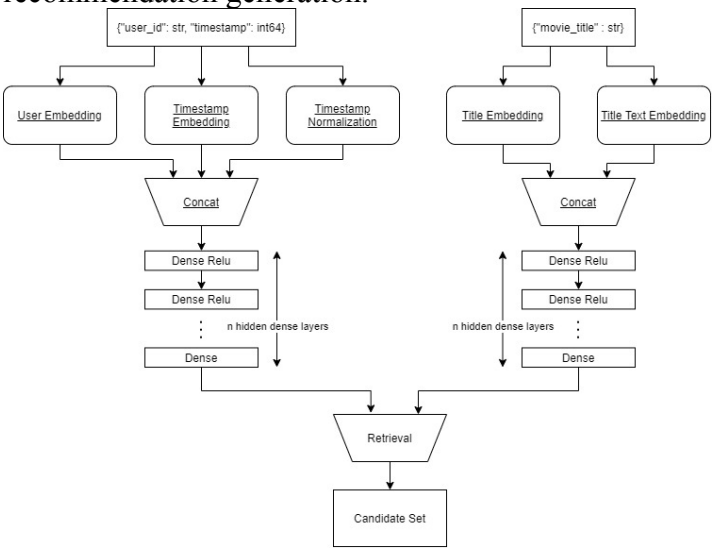


Figure 2: Model structure

2.3 Evaluation

The project was to be evaluated on three metrics to determine success. First, the project's performance was evaluated on the benchmark dataset, Movielens 100K. The performance metric used was Top-K Categorical Accuracy, which computes, as a percentage, how often targets are in the top K recommendations. Scores were computed for K values of 10, 50, and 100.

Dataset	*K=10*	*K=50*	*K=100*
Movielens 100K	1.10%	13.06%	27.56%

Table 1: Top-K Categorical Accuracy of model

Next, it evaluated on usability, which we defined as ease of use, clarity, and versatility of the library. A survey was conducted amongst our peers by providing them with only the library and it's documentation.

Finally the project's impact and value was to be evaluated by the quantity of users the library would have at time of publishing.

3. RESULTS AND DISCUSSION

For performance, the Top-K Categorical Accuracies achieved by the model on the Movielens 100K Dataset are summarized in Table 1. This is on average 233% better than that of a comparable non-deep recommendation model from early in development as seen in Table 2.

Dataset	*K=10*	*K=50*	*K=100*
Movielens 100K	0.30%	7.09%	18.39%

Table 2:Top-K Categorical Accuracy of non-deep model

For usability, the average ratings for ease of use, clarity and versatility were 8.7, 8.4 and 7.3, respectively. Valuable feedback was provided by the survey participants on regions in which to focus improvements going forwards.

For impact and value, due to setbacks, we were unable to publish the library in advance of publishing. However, we received very positive feedback from the survey participants, which suggests that this library will create impact and be a valuable tool moving forwards.

4. CONCLUSIONS AND FUTURE WORK

This project brought to light the ubiquity of recommendation systems throughout the modern UX will serve as a powerful tool to increase their accessibility.

Moving forward, to improve the performance of the library we would implement processing of more contextual information which would produce more valuable insights. Additionally, moving away from the current embedding-based model could improve model performance on unseen data, improving generalization. Also we hope to broaden support of the library to graph datasets, which would encompass many social network and relational recommendation problems.

Secondly, further work is required on improving usability. Incorporating feedback from the survey participants, creating more detailed and example driven documentation would improve learning efficacy. Additionally, broadening functionality in evaluation and exposing more parameters would improve clarity and versatility.

Finally, the library needs to be prepped for publishing to PyPI, which will predominantly be tasks like code cleanup and the addition of a testing framework.

In summary, we put together a library which implements a powerful recommendation model on your dataset in a matter of lines. Our work opens the door for all amateur developers to tailor their user experiences and gives them the tools to create impact with their work.

REFERENCES

[1] L. Gong, ‘Can web-based recommendation systems afford deep models: a context-based approach for efficient model-based reasoning’, in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, New York, NY, USA, May 2004, pp. 89–93, doi: 10.1145/1013367.1013383.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, ‘Deep Learning based Recommender System: A Survey and New Perspectives’, *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Feb. 2019, doi: 10.1145/3285029.

[3] ‘TensorFlow Recommenders’. https://www.tensorflow.org/recommenders (accessed Mar. 18, 2021).

[4] J. Kirk, ‘Tensorrec’, *GitHub*. https://github.com/jfkirk/tensorrec (accessed Mar. 18, 2021).

[5] X. Yi *et al.*, ‘Sampling-bias-corrected neural modeling for large corpus item recommendations’, in *Proceedings of the 13th ACM Conference on Recommender Systems*, Copenhagen Denmark, Sep. 2019, pp. 269–277, doi: 10.1145/3298689.3346996.

Hospital Scheduling

Ariana Bakhtyari[1], Ben Graham[2], Nayana Menon[3], Anshul Pattoo[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada.

1 e-mail: 17ab70@queensu.ca
2 e-mail: 19bng@queensu.ca
3 e-mail: 17nm51@queensu.ca
4 e-mail: anshul.pattoo@queensu.ca

Abstract: Our goal was to use a decision tree algorithm to extract specific rules from a given hospital staff schedule. With scheduling being costly and time consuming, our algorithm extracts specific decision rules to ease the process. Hospital scheduling is a crucial aspect of daily planning. A company has developed a program that will take a set of rules from a client and create a schedule, but they were unable to devise a method to automatically extract the rules from a given schedule. The company typically spends hours communicating back and forth with clients, costing time and money. Their biggest challenge is for the client to remember all the workplace rules, vacations, and personal conflicts for their employees. The program we created solves this by extracting key rules for input. We designed a Gini impurity based decision tree to predict what job any given employee would perform, based on historical scheduling data. We then developed an algorithm to analyze the decision trees output, and extract trends based on the tree's decision criteria at each node. Finally, the results of the analysis are returned in a way that is easily interpreted by a human. Our model performed with an accuracy of 81%. This will provide the scheduling algorithm with accurate and reliable rules, helping ease the process in predicting future schedules for the next respective time period.

1. INTRODUCTION

1.1 Motivation

In recent years, the importance of accurate and effective hospital schedules has become evident. Developing these schedules for hospital employees is exhaustive and complex and directly affects hospital organizational structure [1]. Consistency must be maintained within all divisions of the hospital to ensure smooth operations. This illustrates the importance of a scheduling system that is able to do its work effectively and far in advance to meet all the necessary requirements for employees and hospital operations [1].

Additionally, hospital schedules have a significant impact on patient wait times since healthcare capacity must match patient demand [2]. Wait times continue to increase due to imbalances in the supply and demand chains of hospitals, and this burdens patients and reduces medical care quality [3]. Software has been developed to create the required schedule, but it is not yet possible to extract the rules required to create it. This project aims to simplify the scheduling process through the use of a rule extraction algorithm. Machine learning techniques were used to develop an algorithm that could extract key rules for employee schedules from previous schedule data. This will allow for simplified input into a scheduling system and ease the scheduling process.

1.2 Related Works

Rule extraction is a common machine learning process, but it is mostly applied to other models. One such example is research that has been done to linguistically interpret rules extracted from numerical data for pattern classification and use in genetic modelling [4]. This research, though in a different field, determined that simple rules can be found through classification approaches. After determining a

target rule, a classification algorithm was used for extraction of data [4]. Research has also been done on the ability of machine learning networks to conduct rule extraction, as a more complex form of classification [5]. By developing specific weights for layers of the network, a system can be established to obtain a higher accuracy output [5]. These rule extraction methods and the research associated with them were the main basis for our design process.

1.3 Problem Definition

This project aims to simplify the issue of outdated programs and inefficient scheduling systems by developing a classification algorithm to perform rule extraction for efficient scheduling practices. Being less complex and work intensive than methods like memetic algorithms and statistical analysis, this methodology will allow for an easily adaptable program to be created. This will allow changes to be made as hospital operations evolve with low effort, and as machine learning libraries and tools are created. The resultant program will extract rules from employee input that involve their requirements for a schedule using a decision tree model. These rules can then be coded and input into a schedule using a separate program.

2. METHODOLOGY

2.1 Dataset Manipulations

Start Date	End Date	Job	Member	Worked Hours
2021-01-01	2021-01-01	Tester A	Joe	8
2021-01-01	2021-01-01	Tester B	Bart	8
2021-01-02	2021-01-02	Tester A	Zeta	8
2021-01-02	2021-01-02	Tester B	Hathy	8
…	…	…	…	…

Table 1: Initial data set.

A sample of the initial data set proved from the company can be seen in Table 1. The data set is an example of a typical hospital schedule and provides the start and end dates of each shift, the job which the employee worked on that shift, the name of the employee, and the length of the shift. Since this base dataset was limited and only included five attributes and less than 200 entries, data augmentation was performed on the set. This process involved first replacing all date attributes with an integer representing the day of week, since this provides more insight into scheduling trends than the calendar date itself. Second, categorical data such as name and job were encoded with a numeric placeholder to allow the model to use these attributes. Finally, the dataset was interpolated and additional attributes including number of staff taking time off, and cumulative number of shifts worked were added.

2.2 Model Development

The proposed solution uses a decision tree classifier model as it allows for the model to be easily interpreted by analysing the criteria on which the tree splits the data at each node. The tree was trained separately to classify data set entities based on a variety of different attributes used as labels for the classification, including *Day of the Week, Member,* and *Job Worked*. It was found that the model performed with different accuracies in each case, and the results are summarized in Table 2.

Target Attribute	Accuracy
Day of the week	40%
Member	62%
Job Worked	81%

Table 2: Model accuracy with different data set attributes used as labels.

Using *Job Worked* as the label in the classification model was selected as it provides the highest accuracy. Running the trained decision tree on the augmented data set provided the text representation seen in Figure 1, from which trends and rules were extracted.

3. RESULTS AND DISCUSSION

Our final model performed well, with an accuracy of 81%. We were tasked to produce decision rules for a set of eight sample staff members — Joe, Bart, Zeta, Hathy, Yolanda, Beiko, Bob, and Sally — and two sample jobs — Tester A and Tester B. A few insights

from our decision tree for this particular example are as follows:

1. Joe, Bart, and Zeta tend to work as Tester A.
2. Hathy, Yolanda, Beiko, Bob, and Sally tend to work as Tester B.
3. Yolanda, Beiko, Bob, and Sally tend to work as Tester A.

Figure 1 shows a visual representation of the pertinent decision tree produced.

```
EXPORTED TREE:
 |--- Member <= 4.50
|    |--- Member <= 3.50
|    |    |--- Member <= 2.50
|    |    |    |--- class: 1
|    |    |--- Member >  2.50
|    |    |    |--- class: 2
|    |--- Member >  3.50
|    |    |--- class: 1
|--- Member >  4.50
|    |--- Start Date <= 1.50
|    |    |--- Bob's B Shifts <= 3.50
|    |    |    |--- class: 2
|    |    |--- Bob's B Shifts >  3.50
|    |    |    |--- class: 2
|    |--- Start Date >  1.50
|    |    |--- Bart's A Shifts <= 6.00
|    |    |    |--- class: 2
|    |    |--- Bart's A Shifts >  6.00
|    |    |    |--- class: 2
```

Figure 1: Outputted Decision Tree.

Over the course of the project, the team discovered that the dataset was best suited for a prediction attribute of *Job Worked* by a particular member (i.e. Tester A or Tester B). Our work could be further improved by converting the decision tree into an output that others would understand. It could also be improved with a larger dataset, both in terms of number of records and attributes. With a larger dataset, we can conduct prediction on several attributes and produce a decision tree which accounts for a variety of factors that might be unbeknownst to model designers.

The confusion matrix for the model's performance is seen in Figure 2.

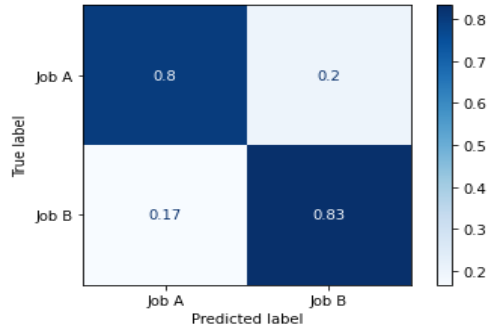


Figure 2: Confusion Matrix.

The confusion matrix indicates that for both output labels, our model performed well. The model was not more predictive of one particular label over another.

4. CONCLUSIONS AND FUTURE WORK

Thus far the team has been able to create a decision tree algorithm model with 81% accuracy. Data preparation, proper attribute creation, and using the Gini-impurity test allowed for the model to extract specific rules from the given schedule.

Many improvements can be made to the model. One modification is to allow the model to handle larger datasets with more variables and still output the same level of accuracy. Furthermore, creating a general code to convert the output of the tree to a coherent set of rules would allow for others to grasp a better understanding of our model.

REFERENCES (IEEE format)

[1] G. Vanden Berghe, "An advanced model and novel meta-heuristic solution methods to personnel scheduling in healthcare," Jun. 2002, Accessed: Mar. 18, 2021. [Online]. Available: https://lirias.kuleuven.be/1675889.

[2] X. Qu, R. L. Rardin, J. A. S. Williams, and D. R. Willis, "Matching daily healthcare provider capacity to demand in advanced access scheduling systems," *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 812–826, Dec. 2007, doi: 10.1016/j.ejor.2006.10.003.

[3] Seattle Children's Hospital *et al.*, "Innovation and Best Practices in Health Care Scheduling," *NAM Perspect.*, vol. 5, no. 2, Feb. 2015, doi: 10.31478/201502g.

[4] H. Ishibuchi, T. Nakashima, and T. Murata, "Three-objective genetics-based machine learning for linguistic rule extraction," *Inf. Sci.*, vol. 136, no. 1, pp. 109–133, Aug. 2001, doi: 10.1016/S0020-0255(01)00144-X.

[5] J. Denker *et al.*, "Large Automatic Learning, Rule Extraction, and Generalisation," Mar. 2021.

Human Activity Recognition in Sports

Luca Trombetta[1], Nathan Kowal[2], Travis Cossarini[3]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: 17lt21@queensu.ca
2 e-mail: nathan.kowal@queensu.ca
3 e-mail: travis.cossarini@queensu.ca

Abstract: *The intention of this project was to apply machine learning on data collected from an Apple Watch to predict various human activities. The results provide a stepping-stone to bringing activity recognition by smart watches to being applied in sports analysis. There were five different activities that data was collected for including jumping jacks, walking, baseball swings, basketball dribbling and basketball shooting. The data was analyzed to look for potential ways of differentiating these activities. To approach the problem, a long short-term memory model and random forest model were developed to predict the recorded activities. The results from each model were compared and had promising measurements of accuracy of over 90%.*

1. INTRODUCTION

1.1 Motivation

Data analysis in sports, typically referred to as "sports analytics", has started to play an integral role for many professional teams and will soon reach an estimated market size of $4 billion [1]. Data collection can help teams win games, decide which players to roster and increase their consumer engagement [1]. In 2015, Major League Baseball implemented a tracking technology known as Statcast, which uses a radar and camera system to analyze player and ball movements at 20,000 frames per second [2]. However, statistics can only take teams so far. While Statcast is able to analyze player speed, position, and distance from a particular location, it cannot access precise player movements [2]. The next logical step would be human activity recognition (HAR) using wearable technology, which would allow for more acute analysis of player activities.

Thus, it was set out to develop a model that will be able to identify various activities with the purpose of application to sports analysis. This model will take input data from wearable technology, opening the door for live data collection and real-time breakdown during training and games.

1.2 Related Works

Other researchers have previously used HAR to analyze sports. Zhuang and Xue used a sliding window approach to segment data and a convolutional neural network (CNN) to analyze sports data in two classifications – non-periodic activity with complex motion states and weakly-periodic activity with complex motion states. Their model preformed with an average recall of about 95%, indicating the success of the sliding-window approach. For future work, they recommended smartwatch implementation of the model to achieve live results for real-world applications [3].

In another application, Hendry et al. worked on HAR as a way to analyze the movements of ballet dancers in an attempt to limit the development of musculoskeletal pain disorders. Their data was manually annotated to classify specific movements into classes before being put into a fixed window of 1 second with a 75% overlap. Using a CNN, their model preformed with 97.8% accuracy at the primary level of classification. Self-identified gaps in their research included using only female dancers and limiting HAR recognition to simple jumps and leg lifting [4].

1.3 Problem Definition

As outlined in the above sections, HAR is becoming a very popular area of research in sports analysis due to its applications in performance improvements and athlete health. This is not the first project that will attempt to classify sport activities. With the knowledge of past research done in the field, the team was able to gain a good understanding of what is possible to accomplish within the given time period. Additionally, this work allowed for essential research into the best type of data collection process and machine learning model.

Based on prior team knowledge of artificial intelligence and machine learning, it was decided that a realistic project goal would be to implement and compare two types of models. Instead of sourcing a dataset from another source, the team agreed that it was feasible to collect and process data ourselves. It was decided that creating a front-end application for real-time model feedback was outside of the scope of the project, given time and absence of prior knowledge in Swift programming.

2. METHODOLOGY

2.1 Dataset Generation

Data was collected using an Apple Watch gyroscope and accelerometer. To access this data from the Apple Watch, an app called Motion Collector was used. It was sourced from GitHub and was originally created by Aleksei Degtiarev [5]. About five minutes of data was collected for each of the five chosen classifications: jumping jacks, walking, baseball, basketball shooting, and basketball dribbling.

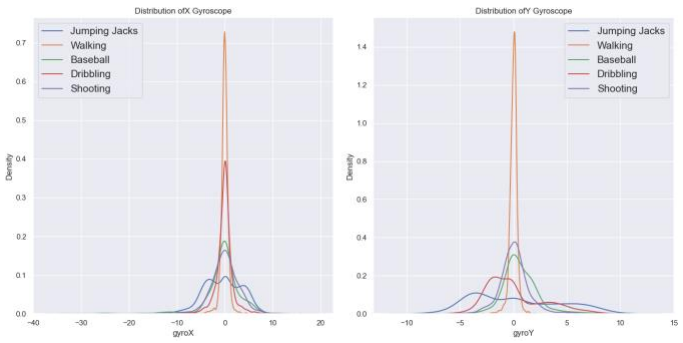


Figure 1: Example of density plots for the X and Y signals of the gyroscope.

Once data was collected, it required preliminary analysis and preprocessing for model implementation. Analysis was done by density plots for each activity and sensor (see Figure 1). In the case of incomplete or missing data, more data was simply collected to replace the incomplete data. After analysis, the data was segmented into two second windows with 50% overlap. Then, a feature set of 27 features was created from the windowed data. Finally, it was divided into training data and testing data to be used on the model, with 25% reserved for testing.

2.2 Model Selection and Development

The proposed solution was to develop a long short-term memory (LSTM) model capable of identifying human activity from the data collected by the Apple Watch. This model was chosen because it of its power for predicting based on time series data. It was developed using the Keras library within TensorFlow. Additionally, a random forest (RF) model was implemented from the Scikit-Learn library for comparison with the LSTM.

The LSTM model was trained on the windowed data set, while the RF was trained on the feature set. The results of both models were determined based on the accuracy of predictions against the test data sets.

2.3 Additional Analysis

To optimize the performance of each model hyper-parameter tuning was carried out on each model. To tune the LSTM a Python library called Hyperas was used to determine optimal output space dimensions, and dropout coefficients for each layer of the model. The optimization of the random forest was performed with the Scikit-Learn library to choose the number of decision trees for the model.

3. RESULTS AND DISCUSSION

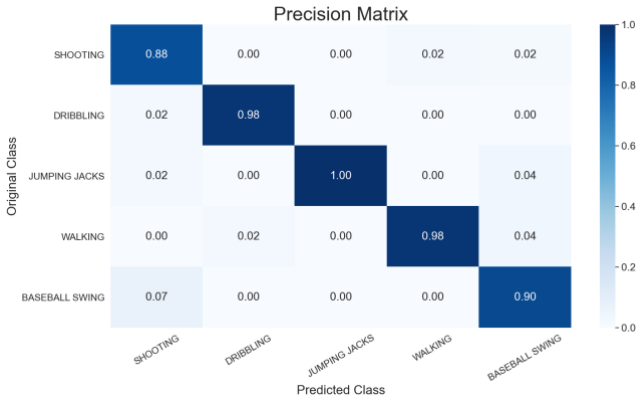


Figure 2: Precision Matrix of LSTM

Figure 2 above shows the precision of the LSTM model on the test data set after hyper-parameterization. The model was evaluated over eight epochs and struggled to differentiate the movements of shooting and swinging a bat. This is likely due to the similarities between these two motions leading to similar data windows from the Apple Watch accelerometer and gyroscope.

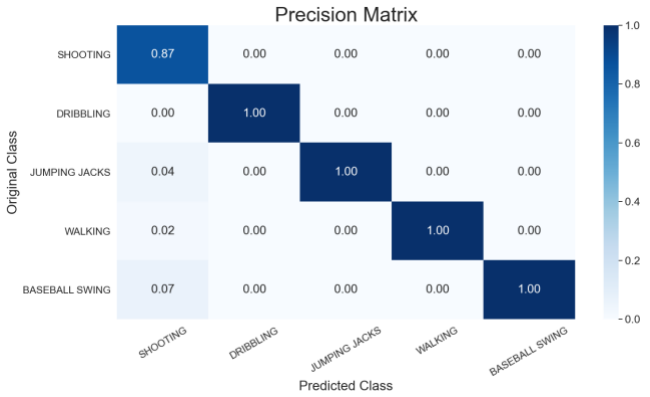


Figure 3: Precision Matrix of Random Forest

Figure 3 displays the precision of the random forest model when run on the test data set. After hyper-parameterization, it was determined that the optimal forest size contained fifty trees. This model performed very strongly in recognizing four of the five human activities, but similarly to the LSTM model, struggled to recognize shooting. Once again, this is likely due to similarities with other motions.

Model	*Accuracy*
LSTM	94.44 %
RF	96.32 %

Table 1: Results of the evaluated models

Table 1 summarizes the results from each model. The LSTM model, which was originally expected to be the stronger model, was outperformed by the random forest model. This is likely due to the set of 27 features developed for the RF model which allowed for more correlation between the data to be found. However, both results have quite high accuracy and exceed the original project goals of 80%.

4. CONCLUSIONS AND FUTURE WORK

The ability to correctly predict movements based only off the accelerometer and gyroscope measurements from an Apple Watch is quite promising and has possible applications in the realm of sports analysis.

In the future, there are several steps to be taken to improve the results of this project. This most immediate step would be to implement an iOS interface allowing for real-time recognition of the watch wearers activities. This an essential component of being able to apply this project in sports analysis. In addition to real-time recognition, there must be more classes introduced to each model. Currently the five classes cover a diverse number of sports, but it would be beneficial to introduce various motions from the same sport, once again brining the project closer to applications in sports analysis.

REFERENCES

[1] A. Ricky. “How Data Analysis In Sports Is Changing The Game.” Forbes. https://www.forbes.com/sites/forbestechcouncil/2019/01/31/how-data-analysis-in-sports-is-changing-the-game/?sh=b602d293f7b4 (accessed Mar. 15, 2021).

[2] “Statcast | Glossary | MLB.com,” MLB.com. https://www.mlb.com/glossary/statcast (accessed Mar. 15, 2021).

[3] Z. Zhuang and Y. Xue, “Sport-Related Human Activity Detection and Recognition Using a Smartwatch,” *Sensors*, vol. 19, no. 22, Art. no. 22, Jan. 2019, doi: 10.3390/s19225001.

[4] D. Hendry, K. Chai, A. Campbell, L. Hopper, P. O’Sullivan, and L. Straker, “Development of a Human Activity Recognition System for Ballet Tasks,” *Sports Medicine - Open*, vol. 6, no. 1, p. 10, Feb. 2020, doi: 10.1186/s40798-020-0237-5.

[5] *MotionCollector*. (2018), A. Degtiarev. Accessed: Nov. 21, 2020. [Online]. Available: https://github.com/degtiarev/MotionCollector

Interview Confidence Scoring

Connor Winters[1], Camila Izaquierdo[2], Ryan Turnbull[3], Ana Pleava[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: c.winters@queensu.ca
2 e-mail: 18cni@queensu.ca
3 e-mail: 18rht1@queensu.ca
4 e-mail: ana_pleava81@outlook.com

Abstract: *For job seekers around the world, the recruiting process can be a daunting and nerve-raking endeavour, particularly when it comes time for an interview. To better prepare jobseekers, the authors have partnered with a Waterloo-based start-up, that provides feedback on mock interview tailored to the role(s) they are applying for. Evaluating an interview is often very subjective, with a successful interview incorporating a wide range of factors, including accurate content, engaging delivery all supported by appropriate non-verbal communication. The complexity and intricacy of an interview makes it difficult to provide consistent and scalable feedback. As such an AI solution was developed, that evaluates an interviewees confidence to provide feedback based on their non-verbal communication. With a computer vision model capable of scoring confidence in a mock-interview, it is possible to provide feedback to users, while simultaneously ensuring scores are consistent and replicable.*

1. INTRODUCTION

1.1 Motivation

For job seekers around the world, the recruiting process can be a daunting and nerve-raking endeavor, particularly when it comes time for an interview.

Interview preparation for most typically consists of doing research on the company and preparing for behavioral or technical questions. One aspect of preparation that is often overlooked in this process is the delivery of content and non-verbal communication.

Since interview preparation is often done individually, people tend to focus more on the content of their responses rather than the delivery. As a result, despite answering questions correctly, candidates may be overlooked due to poor engagement and delivery. The complexity and intricacy of an interview makes it difficult to provide consistent and scalable feedback.

As such an AI solution has been developed, that evaluates an interviewees confidence to provide feedback based on their non-verbal communication.

1.2 Related Works

At the moment, limited tools exist with the focus of providing feedback on nonverbal communication. One similar platform that offers interview preparation is 'Big Interview'. The platform offers hands-on practice with mock interviews tailored to the specific industry, job, and experience level. This platform emphasizes heavily on the content of the response rather than the execution and delivery.

1.3 Problem Definition

To better prepare job seekers, the authors have partnered with a Waterloo-based start-up that provides feedback on mock interview tailored to the role(s) an individual is applying for. At the moment, the Ace platform allows users to sign up and selects their desired field to prep for. The recorded interview is

then evaluated by the Ace founder on different criteria and feedback is then returned to users with actionable insights to improve their interview skills. The key problem here is that this manual process severely limits Ace's ability to scale their platform. Additionally, this manual process may result in unintentional bias and inconsistent scoring.

With a computer vision model capable of scoring confidence in a mock-interview, it is possible to the ability to provide feedback to their clients, while simultaneously ensuring scores are consistent and replicable. For the partner, this is an important step in developing a scalable platform while numerous extensions of this application emerge, particularly for firms looking to reduce bias and human capital requirements for their own hiring processes.

2. METHODOLOGY

From preliminary discussions with the client, it was identified that the video analysis portion of the interview process would be optimized. To analyze the video, two solutions were explored.

2.1 Facial Point Landmarks

The first consisted of creating a model to analyze the movement of facial data points. The data collected from these points would then be filtered through the scoring program to give feedback. Facial landmarks are classified using a pre-trained ensemble of regression trees leveraging Python libraries including OpenCV, dlib, and TensorFlow.

2.2 Facial Emotion Recognition

Another solution that was researched is the Facial Emotion Recognition (FER) model. This Convolutional Neural Network (CNN) was developed using the Python libraries OpenCV and TensorFlow. Using the Haar Cascade classifier and a dataset of over seven thousand emotion labelled images, the model can identify facial expressions in a video stream. The corresponding emotions are shown in real-time above the user's head.

2.3 Model Analysis

The solutions were combined to create a dually functional model that would output two forms of data, as seen in figures 1 and 2 below. The data collected from the joint model is then inputted into the result scoring program to summarize feedback for the user as seen in figures 3 and 4 below.

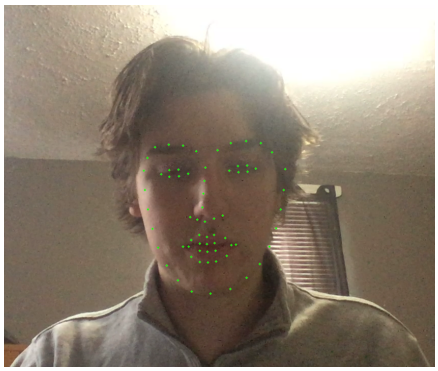


Figure 1: Facial Landmark Detection

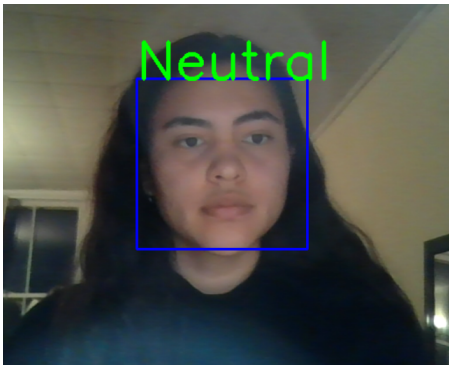


Figure 2: Emotion Recognition

3. RESULTS AND DISCUSSION

The first component of the solution involving facial landmark detection, leverages a pre-trained ensemble of regression trees outputting the x and y coordinates of each facial landmark in a given frame of a recording interview. The model is capable of identifying the correct pixel location for each facial landmark, to the extent a human would be able to accomplish.

The second component of the solution includes a 7-block CNN, with each block incorporating batch normalization and max pooling. The first block includes 32 filters, growing to 256 filters in the final block. Following the convolutional layers, the model includes 3 dense layers followed a output layer with 5 outputs nodes.

	Time	Outline 1	Outline 2	Outline 3	Outline 4	Outline 5	Outline 6	Outline 7	Outline 8	Outline 9	Outline 10
0	0.801866	[461, 158]	[463, 187]	[466, 218]	[474, 247]	[490, 272]	[511, 294]	[536, 312]	[562, 325]	[588, 327]	[608, 321]
1	0.988138	[458, 153]	[459, 184]	[463, 215]	[471, 244]	[487, 270]	[509, 292]	[534, 310]	[560, 324]	[585, 327]	[605, 321]
2	1.126659	[468, 156]	[470, 184]	[473, 214]	[480, 242]	[495, 268]	[515, 289]	[539, 306]	[565, 319]	[591, 321]	[611, 317]
3	1.306186	[477, 148]	[481, 180]	[487, 211]	[497, 240]	[515, 265]	[539, 286]	[564, 303]	[591, 314]	[616, 315]	[635, 307]
4	1.490420	[482, 143]	[486, 175]	[491, 206]	[501, 235]	[518, 261]	[542, 282]	[567, 299]	[595, 312]	[620, 313]	[639, 305]
5	1.691345	[490, 144]	[493, 175]	[498, 206]	[507, 235]	[524, 260]	[547, 281]	[571, 298]	[598, 311]	[623, 313]	[641, 305]
6	1.884796	[-6, 348]	[-8, 359]	[-8, 371]	[-7, 384]	[-5, 397]	[-4, 411]	[-3, 424]	[1, 436]	[15, 442]	[35, 441]
7	3.447637	[489, 144]	[492, 175]	[497, 206]	[505, 235]	[522, 260]	[545, 280]	[568, 298]	[594, 312]	[619, 315]	[638, 308]
8	3.453966	[-16, 345]	[-18, 356]	[-18, 369]	[-16, 382]	[-15, 395]	[-12, 408]	[-8, 421]	[-2, 433]	[12, 438]	[32, 437]
9	3.542166	[485, 149]	[489, 180]	[493, 211]	[502, 240]	[519, 265]	[543, 285]	[567, 302]	[594, 315]	[620, 318]	[639, 310]
10	3.549325	[483, 152]	[487, 182]	[492, 213]	[501, 241]	[518, 265]	[541, 285]	[566, 301]	[593, 314]	[618, 316]	[638, 309]

Figure 1: Results of the first 10 facial landmarks in the first ten intervals of the video.

```
{'Angry': 19, 'Happy': 222, 'Neutral': 65, 'Sad': 48, 'Surprise': 0}
```

Figure 2: Results of the total facial emotions during the video.

The classification report of the emotion recognition CNN model is summarized in Table 1

Emotion	*Precision*	*Recall*
Angry	0.53	0.59
Happy	0.79	0.86
Neutral	0.32	0.22
Sad	0.56	0.52
Surprise	0.69	0.83

Table 2: Emotion Recognition Classification Report

While the overall accuracy is just over 60% the model's tendency is to classify emotions as neutral, resulting in only expressive emotions to be classified beyond neutral.

The final component of the solution is tracking the output of the two models to provide feedback to users. With the facial landmark model, the *x* and *y* coordinates are recorded for each frame or about 22 times per second. After the coordinates are identified the total displacement is calculated. Leveraging academic research in the neuroscience field, it was found that increased levels of movements are linked to nervousness and anxiety, which can be provided as relevant feedback to users.

The emotion recognition model tracks the classified emotion for each frame of a recorded interview and then finds the proportion of each emotion throughout the interview to use in the scoring algorithm.

4. CONCLUSIONS AND FUTURE WORK

With the implementation of the emotion recognition, and facial detection models, the model was able to successfully gather some key data on an interviewee during an interview. However, the data is not perfect, and steps need to be taken to analyze the data properly and apply a proper score to the interview.

Currently the emotion recognition model is only 60-70% accurate and has a tendency to classify emotions as neutral. To improve its functionality in the grading process, a more accurate model is beneficial, alongside a more diverse set of emotions to classify. This will be accomplished by expanding the training data used, and by exploring methods to improve the classification accuracy such as ensemble methods.

Additionally, although the movement of key facial landmarks are tracked over time, it has not implemented this into the scoring. To optimize the scoring system, the movement of facial features will need to be assessed over time and with the findings implemented into the scoring algorithm. Fully incorporating the aforementioned improvements will make the grading system much more substantial and accurate.

Finally, the functionality of tailoring feedback to the desired role of a particular candidate, as different roles may be more receptive to different emotions displayed by candidates.

Product Classification for E-commerce

Joseph Grosso[1], Inika Chikarmane[2], Daisy Dan[3], Mengyang Liu[4], Jose Luis Mangubat[5]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 15jjg6@queensu.ca
2 inika.chikarmane@queensu.ca
3 18cd11@queensu.ca
4 19ml49@queensu.ca
5 17bm1@queensu.ca

Abstract: *Product classification is extremely important for e-commerce companies for the customer experience, but it can be hard to get it right when trying to classify hundreds of thousands of products from disparate sources. Using machine learning and Natural Language Processing techniques, the process of classifying new items into a product category can be automated. With these tools, we were able to create a product classifier that separated products in the Loblaws product catalogue into their correct category at an accuracy rate of ~90%. This system could improve Loblaws manually constructed product categorizations and many aspects of the business and customer experience.*

1. INTRODUCTION

1.1 Motivation

Online shopping is a fact of life for most consumers today, especially during the COVID-19 pandemic. To run a successful online business, a company must optimize the user experience, provide relevant search results for desired products, and ensure that the site can be found easily through search engines like Google [1]. This can be difficult when an online shop has to organize thousands of products from different vendors.

All of these issues can be helped by using product categorization. By classifying products into a taxonomy, the platform can associate products with keywords and better organize their products for search. This is key for improving the customer experience.

Product categorization becomes hard to manually maintain as the number of products a company carries expands quickly. Classifying new products becomes laborious and requires in-depth knowledge of the product hierarchy. Machine learning classification techniques can help with product classification, automating insertion of new products into product taxonomies.

1.2 Problem Definition

The Canadian grocery chain Loblaws manages thousands of products in their catalogue. Each of these products need to be categorized into Loblaws custom product taxonomy. It's extremely important that products are categorized correctly, because a mistake in classification is hard to find and can cause errors for the business and can make products harder to find for customers on Loblaw's online platform.

Using features associated with each product, like a products title and description, we aim to create a machine learning model to classify products in the Loblaws catalogue, and make predictions on brand new products that have no prior sales history. To achieve this, we must use techniques from Natural Language Processing (NLP) to parse the unstructured data in the product titles and descriptions into a form that is useful for our models. We must also take into account the hierarchical structure of our data when considering our model's accuracy and other metrics to assess our model's performance.

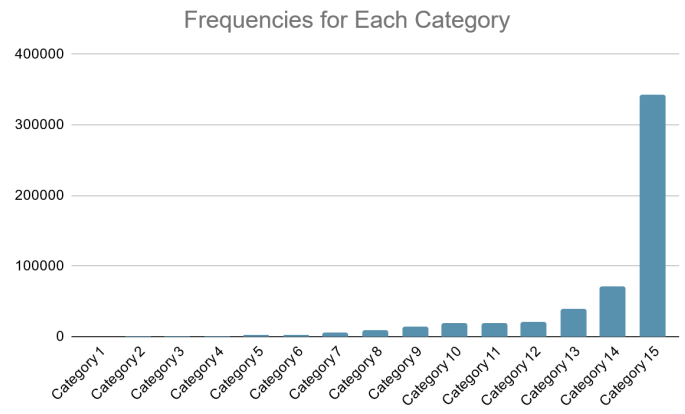


Figure 1: A bar graph showing the imbalance in the dataset between different product categories.

2. METHODOLOGY

The online branch of Loblaws, Loblaw Digital, presented our team with a product catalog of over 120,000+ products. Each product in the dataset had a title, a product description, a unique identifier, and an MCHID, which represented where the product belonged in the product taxonomy.

articleID	MCHID	NAME	Description
20433433_AB	M12345678	Spam Bacon	Ready to eat - cold or hot

Table 1: An example data point from our dataset ofLoblaws products.

For the majority of our models used for our problem, we needed to parse the product descriptions using NLP techniques to allow for machine learning models to train on the information that they are given. This process starts with standard text pre-processing steps, which helped in removing noise from our data. These processing steps included merging the titles and descriptions into a single text, removing unhelpful strings from the text (ie. stop words like "the" and "a", HTML tags, special characters), and applying lemmatization to the tokens (replacing inflected forms of a word so they can be analysed as a single item, ie changing playing, plays, or played to play).

After cleaning the data, we then needed to convert our data into numerical features that a machine learning model could understand. For this, we used the Bag-of-Words (BoW) approach to convert the text into a vector which represented the number of times a word had been used in the text [2]. We used this method because it allowed us to simplify the problem down to the simple presence of a word. This method doesn't rely on the local context of the words like Word2Vec or GloVe, which was useful because the text we worked with didn't follow consistent formats. BoW is also very simple for human understanding, which allows us to directly determine the importance of words in our model. We were also able to leverage sklearn's HashingVectorizer to reduce the memory usage of our vectorization algorithm, and to allow our model to allow for our model to add new words to its vocabulary, should the model be re-trained in the future [3].

To begin our analysis, we trained our models on only the highest levels of the hierarchical structure, meaning that we tried to identify the broadest levels of categorization. This meant the team could immediately work on the project without having to deal with the hierarchical structure of the data, but this model would only have value for the team as an exploratory step. The uniqueness of the hierarchical classification meant that commonly used metrics for model performance like accuracy, precision, and recall would not express the true performance of the model. These "flat" evaluation metrics do not capture how some classes have a stronger relationship than others - for example, if our model misclassified a fruit product as a vegetable, this outcome is better than if the model had misclassified the item as a grain product, which is still better than an incorrect classification as a non-food item. To regulate for these outcomes, we used hierarchical precision and hierarchical recall as our metric to judge these deeper models [4].

3. RESULTS AND DISCUSSION

We have trained most of our models on the broadest category as mentioned in the Methodology section. Using our transformed dataset, we trained many machine learning models using SciKit Learn and Keras. We found that three of our models performed the best: Multinomial Logistic Regression and Random Forest from scikit-learn, and Recurrent Neural Network (RNN) using Keras.

Model	Accuracy
Random Forest	**86.07%**
K-Neighbors	82.13%
Naive-Bayes	76.43%
Support Vector Machines	84.26%
Recurrent Neural Network	**91.06%**
Logistic Regression	**85.27%**

Table 2: Results of our models trained on the highest levels of the hierarchy structure, with the most promising models highlighted in bold.

The model with the best accuracy is RNN, with an accuracy of 91.06%. While Logistic Regression is not performing as well as RNN or Random Forest, it offers the benefit of being straightforward and offering an implementation of Kesler's Construction (using softmax regression) that allows for a single model to be trained for multinomial classification, unlike the other methods that use a One-vs-One or One-vs-Rest approach [5]. This method has also been implemented successfully by other data science teams [6], and so we will continue with this model in our testing on the lower levels of the hierarchy.

The performance of these models is promising, showing that we may be able to move these models down to the lowest levels of the product hierarchy successfully. A successful classifier would return the top N most likely classes that a given product belongs to given its name and title, and would classify based on the most granular classification level.

4. CONCLUSIONS AND FUTURE WORK

We have many ideas for how we can improve our machine learning models in future work. Our models will continue to be optimized to classify on the lowest, most granular levels of the hierarchical structure, which is where the most value will come from for the business. In the coming weeks we will be implementing this functionality on the model using the techniques we found effective at higher levels and using our hierarchical performance metrics. We also see value in using oversampling techniques to address the issue of large imbalance between classes in the dataset.

We will also make the model production system ready by wrapping all of our functionality in well-documented classes, and by implementing a server in Flask that will allow any system at Loblaw Digital to make a request to our model for classification.

In conclusion, our group has created a fully fleshed out approach to the problem of product classification for an e-commerce site. With our product, Loblaw Digital will be able to provide a better experience for both their online customers and their vendors. We found that this type of product could bring value to any e-commerce site with a large product catalogue.

REFERENCES

[1] Limarc Ambalina, "What is Product Categorization?", Lionbridge.ai, October 10, 2019

[2] Jason Brownlee, "A Gentle Introduction to the Bag-of-Words Model", Machine Learning Mastery, October 9, 2017

[3] scikit-learn version 0.24.1, (scikit-learn developers), "HashingVectorizer"

[4] Eduardo P. Costa, Ana Carolina Lorena, A. Carvalho, A. Freitas, "A review of performance evaluation measures for hierarchical classifiers", Proceedings of the AAAI 2007 workshop ''Evaluation methods for machine learning", 2007

[5] Kai-Wei Chang, "Seminar: Machine Learning in Natural Language Processing, Lecture 3: Multi-Class Classification", University of California Los Angeles, 2017

[6] Jeet Mehta, "Categorizing Products at Scale", Shopify Engineering, April 30, 2020

Stock Options Volatility Prediction

Alexandre Le Blanc[1], Andrew Brown[2], Smeet Chheda[3], Tanner Dunn[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada.

1 e-mail: a.leblanc@queensu.ca
2 e-mail: 17amb@queensu.ca
3 e-mail: smeet.chheda@queensu.ca
4 e-mail: tanner.dunn@queensu.ca

Abstract: The volatility of the Stock Market is an important factor that should be taken into consideration when trading options. Previous research has shown that using machine learning techniques, predicting volatility can lead to positive returns. Using the CBOE Volatility Index (ticker: VIX), a time-series ARIMA model was used to forecast the volatility of the S&P 500. Using the predictions, options trading strategies were recommended based on the differences between the current and forecasted volatilities. The ARIMA model was able to obtain an AIC of 6665.5, and a MAPE of 7.5% on 14-day forecasts. The findings indicate that by using time-series forecasting, the volatility of the market can be isolated and used to generate greater alpha.

1. INTRODUCTION

1.1 Motivation

The stock market has been a money-making tool for millions of people around the world. Whether it be day trading, investment banking, or simply placing one's savings in a mutual fund, the market has helped countless people achieve financial freedom, or at least get closer to it. This, however, comes with a significant and possibly costly learning curve. Several prominent figures in the finance world have agreed that beating the market is always possible, but not probable [1].

At its foundation, beating the market comes down to being able to look at data, make connections and turn it into useful information. With the vast arsenal of machine learning tools available on the internet, it is quite possible to aggregate this data and make intelligent decisions from it. Nevertheless, determining the direction of future stock price movement is challenging. Using options trading strategies, traders can shift the focus of their analysis from predicting the *direction* of price movement to predicting the *magnitude* of the movement.

1.2 Related Works

A paper published by the engineering faculty at New York University discusses this problem in a similar context [2]. They utilize learning regression methods to predict the realized variance of the S&P 500. By developing algorithms to correct deviation between the VIX and the actual realized variance of the SPX, they were able to better predict it. It is noted that out of money options are more predictable and the implied volatility of calls have more significance in data manipulation.

1.3 Problem Definition

The aforementioned paper demonstrates the viability of using machine learning to formulate accurate predictions about the market. Our exploration, however, takes a slightly different approach.

Options contracts are derivatives of regular securities and present traders with the ability to capitalize on both upward and downward movements in the market. They represent an agreement to purchase or sell shares of a given stock at a pre-defined price before a set expiry date. Using different contracts simultaneously allows investors to formulate strategies that can mitigate risk. These strategies can be constructed in

such a way where the trader sees a profit if the underlying stock price moves significantly, regardless of direction. Strategies can also be built such that a profit is made if the underlying stock price moves very little. In other words, options trading strategies allow traders to shift their focus from predicting the direction of stock price movement, to predicting the magnitude of the movement, represented by the stock's volatility.

This study analyzes the volatility of the VIX to extract a prediction as to the strength of the market in a given future period. Then a decision-making algorithm outputs an appropriate options trading strategy that reflects the model's view on future volatility.

2. METHODOLOGY

2.1 Data Exploration

The team researched and analyzed the Chicago Board Options Exchange (CBOE) Volatility Index (VIX). The price of the VIX is obtained from the implied volatility of various options contracts belonging to securities in the S&P 500. The data was taken from Yahoo Finance using the "yfinance" API. This API allows users to scrape financial data in a specified time window. The model was designed to analyze the recent volatility of VIX to forecast the future volatility and determine which options trading strategy would yield the greatest returns.

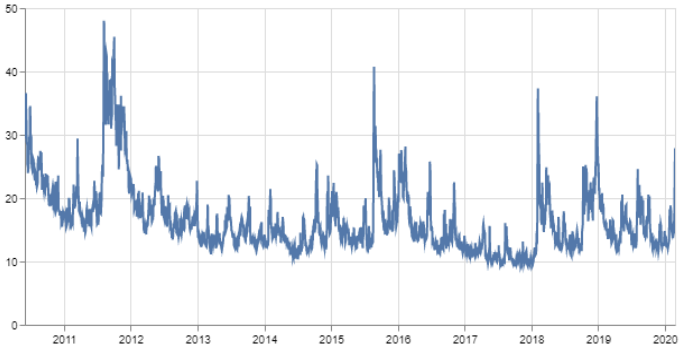


Figure 1: VIX Close Price 2010 - 2021

Figure 1 above displays the VIX closing price since 2010. This chart essentially represents what professionals think of the volatility of the S&P 500. The dataset's features are 'Date', 'Open', 'High', 'Low', 'Close', and 'Adjusted Close'. The 'Close' price was selected as time-series data to analyze and to be forecasted.

2.2 Proposed Solution

The proposed solution is to use the time-series analysis algorithm known as an Auto Regressive Integrated Moving Average (ARIMA) model to predict the future values of the VIX. ARIMA models are a combination of Auto Regressive (AR), Differencing (I), and Moving Average (MA) models, discussed below:

AR(p): Auto Regressive Model
AR models forecast based on their own lags (a lag simply refers to the previous values in the time-series), utilizing the p^{th} first lags (*i.e.*: an AR(3) model uses the first three lags). The equation for AR is shown below, where Y_{t-1} is the first lag of the series, β_1 is its corresponding coefficient, Y_{t-2} and β_2 are for the second lag, and so on, α is the intercept term, and ϵ_1 is the error term:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_1$$

I(d): Differencing Parameter
The differencing parameter d refers to the number of differencing required to make a time-series dataset stationary (an important requirement for datasets used for forecasting). In this context, differencing refers to subtracting the original time-series by itself shifted backwards by one. The number of differencing required to make a dataset stationary determines the d parameter. The Augmented Dickey-Fuller test was used to determine stationarity.

MA(q): Moving Average Model
MA models base their current forecasts on the q^{th} most recent errors (residuals) of the previous forecasts. The equation for MA is shown below:

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + \cdots + \phi_q \epsilon_{t-q}$$

Where α is once again the constant intercept term, ϵ's are the error terms, and ϕ's are the error terms' constant coefficients.

ARIMA(p,d,q)
An ARIMA model formulates its prediction by combining all of the aforementioned models, that is, by using its own lags and its previous forecasts errors, as shown in the equation below:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \cdots + \beta_p Y_{t-p} + \phi_1 \epsilon_{t-1} + \cdots + \phi_q \epsilon_{t-q} + \epsilon_t$$

Note how this is simply a combination of the AR and MA models' equations.

The optimal *p* and *q* values were determined through visual analysis of autocorrelation function (ACF) plots and partial autocorrelation function (PACF) plots.

2.3 Solution Evaluation

The final portion of the solution is to suggest an options trading strategy. The success of the strategy depends on the volatility of the specified derivative. This means the goal of the ARIMA model is to predict high or low volatility. The team tackled this assignment by comparing the forecasted volatility value to the most recent volatility value. If the value is forecasted significantly higher, a high volatility strategy is recommended, and vice versa. Furthermore, if the volatility is predicted to remain relatively stable, a medium volatility strategy is recommended. An example of this can be seen below in Figure 2. In this case, the ARIMA model forecasted a lower volatility value than the current volatility, thus causing it to suggest a Married Put (low volatility) strategy.

Figure 2: Example Output from ARIMA and Decision Algorithms.

Evaluating the solution was done by having the ARIMA model forecast over a certain number of past data points with known VIX values, *i.e.*: predicting over the most recent 21-days, using all prior data points. The forecasted values were then compared to the actual values to determine its prediction accuracy (discussed below).

3. RESULTS AND DISCUSSION

Using the ARIMA analysis steps described above, it was determined that the optimal *p*, *d*, and *q* ARIMA hyperparameter combinations were (1,0,0) and (0,1,1). To evaluate both models, the following metrics were used: Akaike Information Criterion (AIC), Mean Absolute Percentage Error (MAPE), and Correlation. The AIC is a commonly used metric in statistics to determine how well a model fits the data it is trained on. The AIC favors models which use less parameters in order to avoid over-fitting. Models with low AIC are ideal. The MAPE, as the name suggests, computes the percentage difference between the actual and forecasted time-series' values at each time stamp, and averages these values, returning a value between 0 and 1. Once again, models with lower MAPE scores are favorable. The Correlation metric is simply the correlation between the actual and forecasted data points, where we want the correlation to be high between the two datasets, indicating that the forecasted data resembles the actual, at least in trend. The MAPE and Correlation metrics were selected as they are unaffected by the scale of the data, outputting values in the range [0,1] and [-1,1] respectively. Therefore, the scores can be easily evaluated and compared regardless of the magnitude of the time-series' data.

To obtain the results, we evaluated both models against the VIX dataset with a 14-day and 21-day predictions. Tables 1 and 2 summarize these results:

Accuracy Metrics	*ARIMA(1,0,0)*	*ARIMA(0,1,1)*
AIC	***6665.5***	6704.9
MAPE	***0.07521***	0.11747
Correlation	***0.83650***	-0.83162

Table 1: Summary of various accuracy metrics for ARIMA(1,0,0) and ARIMA(0,1,1) models on the VIX dataset with 14-day forecast periods. Italicized and bolded scores indicate the favorable scores.

Accuracy Metrics	*ARIMA(1,0,0)*	*ARIMA(0,1,1)*
AIC	***6665.5***	6704.9
MAPE	0.09874	***0.09413***
Correlation	***0.45685***	-0.50305

Table 2: Summary of various accuracy metrics for ARIMA(1,0,0) and ARIMA(0,1,1) models on the VIX dataset with 21-day forecast periods. Italicized and bolded scores indicate the favorable scores.

As expected, the ARIMA(1,0,0) performed the best in most cases, with the lowest AIC scores and positive correlation values of 0.84 and 0.46 for 14-day and 21-day forecasts, respectively. This indicates that the ARIMA(1,0,0) best fit the VIX data, and that its forecasts are correlated to the actual values to a certain degree. Although the ARIMA(0,1,1) obtained the best MAPE score for a 21-day lag, its correlations were negative, which is undesirable as it suggests that in most cases, the model's predictions are in the opposite

direction of the actual values, which is evidently a bad trait for the model to have.

Qualitative analysis of the forecast plots, shown in Figures 3 and 4, make it clear that the ARIMA(1,0,0) model outperforms the ARIMA(0,1,1) model as it provides more realistic predictions.

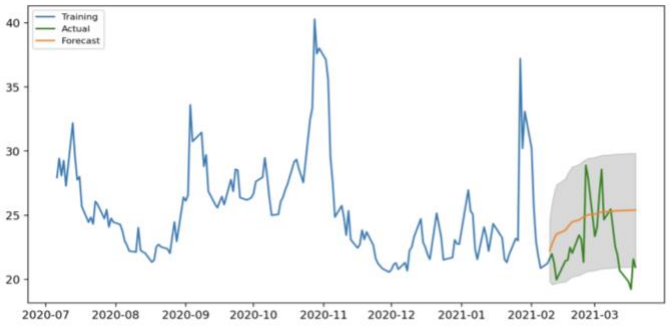


Figure 3: Plot of ARIMA(1,0,0) model on VIX dataset with 28-day lag.

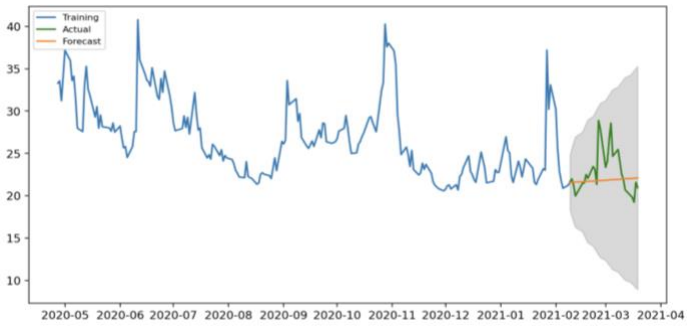


Figure 4: Plot of ARIMA(0,1,1) model on VIX dataset with 28-day lag.

The plots make it clear that the ARIMA(1,0,0) is more suited for the VIX dataset than the ARIMA(0,1,1) model, as the ARIMA(0,1,1) fails to capture even the slightest trend of the actual data in its forecast.

Note that even the favorable ARIMA(1,0,0) model, which accurately forecasts the overall direction of the dataset, is still very simplistic. This is in part a result of the fact that ARIMA models (especially a (1,0,0) model) use very few past data points to make their forecast. For a dataset that has occasional large jumps and dips, this is a major limitation.

4. CONCLUSION AND FUTURE WORK

The team focused on stock volatility prediction using an ARIMA model time-series forecasting. A major focus of time and effort was dedicated to determining the optimal *p*, *d*, and *q* values based on analysis of data stationarity, ACF and PACF plots. Once these values were determined, the optimal ARIMA models were tested against historical VIX data to compare predicted and actual volatility to determine forecasting accuracy. The final solution then implemented a decision-making algorithm which compared its forecasted volatility to the current volatility in order to suggest a favorable options trading strategy for greatest yield.

Throughout the project, it was concluded that ARIMA models are quick and easy forecasting tools that can capture general trends and make reliable predictions on simpler and clean 'vanilla' time-series data. However, it falls short in capturing the finer details in very volatile and noisy datasets, which would be necessary to make profitable bets in the stock market. Thus, if you require predictions which can capture the subtleties of a complex dataset, you may find more success using a different model, such as a recurrent neural network (RNN). As a next step, the team would like to implement an autoregressive RNN, which has the capability of being trained on more data and allows past predicted values to be fed back into the model to refine the predictions at each step.

Another next step for the project would be to run the model against live price data and perform real options trading to get real results on profitability. If the results of these tests are accurate, the scope of this model could be expanded to include different stock indexes. This model has the potential to be powerful if it can predict on a wide variety of indexes, as it could be used to rapidly determine which options trading strategy are best suited in any situation.

5. GITHUB REPOSITORY AND DEMO

https://github.com/andrewmbrown/CUCAI2021_Demo

https://share.streamlit.io/andrewmbrown/cucai2021_demo/main/main.py

6. REFERENCES

[1] A. G. Ribeiro, "Can Regular Investors Beat the Market?," Investopedia, 16 October 2019. [Online]. Available: https://www.investopedia.com/articles/trading/10/beat-the-market.asp. [Accessed 20 March 2021].

[2] P. Carr, L. Wu and Z. Zhang, "Using Machine Learning to Predict Realized Variance," New York University, New York City, 2020.

Stroke Prediction

Sydney Caulfeild[1], Sarah Pak 2[2], Nathanael Yao[3], Hoz Rashid[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1 e-mail: sydney.caulfeild@queensu.ca
2 e-mail: 16shp2@queensu.ca
3 e-mail: 18ny13@queensu.ca
4 e-mail: 16har@queensu.ca

Abstract: *Stroke prevention methods were explored because strokes are the leading cause of death in Canada and the number of deaths is expected to grow over the next decade. As a solution to this problem, a model was created to predict a person's risk of having a stroke based on their physical traits and lifestyles. Current models that use facial recognition to detect strokes provide certain benefits but are less desirable than the proposed stroke prediction model as they cannot detect strokes before they happen. The team utilized a multilayer perceptron model on data containing attributes such as gender, blood pressure, and body mass index to predict a person's likelihood of stroke. The model was able to achieve a 96% accuracy on testing data.*

1. INTRODUCTION

1.1 Motivation

Stroke is the third leading cause of death in Canada. Strokes are caused by the sudden loss of brain function that follows a brain blood vessel blockage or rupture. Its symptoms include loss of sensation, difficulty speaking, vision difficulties, headache, and loss of coordination [1].

The effects of a stroke can range from mild to severe, and among those who survive their stroke, many never fully recover. According to Statistics Canada, approximately 36% of these survivors are left with significant disabilities five years after their stroke and more than 40% require assistance with activities of daily living [1].

In a study conducted in 2015 by the American Heart Association, the number of individuals experiencing the effects of stroke in Canada is projected to increase from 405 000 in 2013 to between 654 000 and 726 000 in 2038 [2].

With the high rates of stroke in Canada and the expected growth in the next decade, the team was motivated to explore methods for stroke prediction and hence prevention.

More specifically, different machine learning models for stroke detection were compared and optimized.

1.2 Related Works

There have been various studies on stroke prediction, including research conducted in 2018 that used images of facial features to determine whether a person has had a stroke [3].

They focused on the expressional asymmetry and mouth askew to make predictions. To classify facial stroke, the Support Vector Machine (SVM), Random Forest (RF), and Bayesian Classifier were adopted as classifiers. The research found that the recognition accuracy of SVM, Random Forest, and Bayes are 100%, 95.45%, and 100%, respectively [3].

1.3 Problem Definition

Models that use facial expressions require pictures to inform a patient if they are having a stroke. Rather than identify a case of stroke, the desired solution will use data on people's lifestyles to state their likelihood of having a stroke. Not only is this data more accessible than pictures, but it can also prevent cases by giving patients their chance of stroke and allowing them to change their unhealthy habits.

2. METHODOLOGY

2.1 Data Generation

Many factors affect one's susceptibility to stroke, including gender, blood pressure, body mass index (BMI), and physical activity [4]. A dataset was sourced from Kaggle as it contains all the major stroke risk factors [5]. The dataset has 11 attributes and a label indicating whether each patient has had a stroke.

A heatmap was used to analyze the correlation between all attributes and stroke. As seen in Figure 1, BMI has the weakest relationship to stroke, however, the performance decreased when the attribute was removed. BMI is also the only attribute containing null values which were all replaced with the BMI attribute's mean.

Figure 1: A heatmap used to visualize the relationship strength among attributes and stroke.

2.2 Model Creation

The selected model for this solution is a multilayer perceptron. Multilayer perceptron's are a feedforward neural network that have an input layer, an output layer, and hidden layers. Neural networks such as multilayer perceptron's are useful for predicting results based on input data because the networks are structured in different layers.

After some testing with other types of models such as an SVC model, the team chose to use a multilayer perceptron because it is effective at handling a wide range of different features. Since the stroke detection data has 11 attributes, the results that the team got from the multilayer perceptron model were better than other models that we tested.

2.3 Additional Analysis

Further modifications were done to the model to optimize the multilayer perceptron model's performance. The number of epochs, nodes, and layers were varied to analyze their impact on the model's behavior. All modifications were evaluated by comparing the model's maximum validation and training scores.

3. RESULTS AND DISCUSSION

Model	*Training*	*Testing*
V1	0.960	0.952
V2	0.960	0.9508
V3	0.9544	0.9615
V4	0.9579	0.9532

Table 1: Results of the different model variations. The training values are the model's accuracy during its training while the testing values are the model's accuracy on the test data.

In the first version of our model, V1, values were normalized using a mapping function that takes in data and outputs an array of unique strings. The multilayer perceptron had a top layer with 10 nodes, a middle layer with 20 nodes, and an output layer with 3 nodes. The model was tested with 150 epochs and a batch size of 30.

The first version of the model demonstrated overfitting after 16 epochs. Therefore, the second version, V2,

had only 30 epochs and another middle layer with 10 nodes. However, as seen in Table 1, the testing accuracy for the second version decreased.

For the third version of our model, V3, the number of nodes in each dense layer of the multilayer perceptron were modified. The number of nodes in the top layer was equal to the number of features, two middle layers were added with 30 nodes and 10 nodes respectively, and an output layer with 3 nodes was used.

In the final version of the model, V4, the number of nodes was changed again for the four dense layers. In the top layer the number of nodes was again equal to the number of features. There were two middle layers with 40 nodes and 30 nodes, and an output layer with 2 nodes.

The third version produced the highest percent accuracy during testing and as such was selected as the final model.

4. CONCLUSIONS AND FUTURE WORK

The team built a model that predicts a person's chance of having a stroke given certain inputs. Using data to assess a person's risk of having a stroke may help assist doctors in identifying people who are at high risk and give them appropriate advice and treatment.

The next steps for the model include testing different learning methods such as Random Forest to determine which learning method is optimal for this dataset. Although a multilayer perceptron model was selected because of its efficiency at handling multiple features, it is important to test other models.

Furthermore, the model may be improved by testing out different methods of data preprocessing to know which data is most useful for predicting a stroke. For instance, other inputs can be removed or modified to see which inputs are important to getting an accurate prediction.

Finally, more data should be collected to construct our own dataset to put our model to the test against current data.

REFERENCES

[1] H. Krueger, J. Koot, R. E. Hall, C. O'Callaghan, M. Bayley and D. Corbett, "Prevalence of Individuals Experiencing the Effects of Stroke in Canada: Trends and Projections," NIH, 2015.

[2] Government of Canada, "Stroke in Canada: Highlights from the Canadian Chronic Disease Surveillance System," 09 12 2019. [Online]. Available: https://www.canada.ca/en/public-health/services/publications/diseases-conditions/stroke-canada-fact-sheet.html. [Accessed 2 March 2021].

[3] C.-Y. Chang, M.-J. Cheng and M. H.-M. Ma, "Application of Machine Learning for Facial Stroke Detection," IEEE, 2018.

[4] A. S. A. "Stroke Risk Factors," 2021. [Online]. Available: https://www.stroke.org/en/about-stroke/stroke-risk-factors. [Accessed 2 February 2021].

[5] F. Soriano, "Stroke Prediction Dataset," 1 January 2021. [Online]. Available: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset?fbclid=IwAR0JNuSXufCZgf16g4Hoiq8mfoULN28r6wtxZ-zXEWx-qWJQttIhDOieC2k. [Accessed 8 January 2021].

Using Depth Information to Improve Object Recognition with Deep Learning

Awni Altabaa[1], Eduard Varshavsky[2], Jada Buchanan[3], Ngoc Bao Han "Mimi" Nguyen[4]

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada.

1 e-mail: awni.altabaa@queensu.ca
2 e-mail: 18ev@queensu.ca
3 e-mail: 16jmb7@queensu.ca
4 e-mail: 19bhnn@queensu.ca

Abstract: *Over the past decade, deep learning has driven great progress in computer vision and 2D image understanding. On the other hand, 3-Dimensional image understanding is still comparatively immature. In recent years, RGB-D cameras combining visual and 3D shape information have become more accessible, enabling progress to be made in the field. In this paper, we test the hypothesis that RGB-D object recognition models can improve on state-of-the-art RGB models and propose a deep learning architecture that leverages the added depth information. Our proposed architecture encodes depth images into a geocentric embedding and makes use of two independent processing streams for the RGB and depth images. We train multiple models to control for and validate the effects of the added depth information. Our best model achieved an accuracy of 70.1% on a dataset of 51 classes.*

1. INTRODUCTION

1.1 Motivation

The extraction of a high-level understanding of three-dimensional (3D) images is a fundamental problem in the field of computer vision. 3D image understanding has applications in areas including remote sensing, mapping, monitoring, autonomous-driving, virtual/augmented reality, and robotics [1]. Thus, models which can autonomously extract high-level information (such as recognizing objects) from 3D images are in high demand.

In past decades, similar to 2D computer vision, research on 3D computer vision often employed classic machine learning methods like Support Vector Machines and Random Forests. [2]. However, with the increase of computational power and availability of data, deep learning has allowed for rapid development in both 2D and 3D computer vision [3].

Our focus is on 3D sensed data in the form of so called RGB-D images. The format consists of a pair of images; a standard RGB image and a depth image. Depth images provide additional information about the 3D structure of the scene, and unlike RGB images, are invariant to lighting and are particularly useful in background separation [4].

1.2 Related Works

One approach to the problem is to simply stack the RGB and depth images generating a 4-channel image and employ existing Convolutional Neural Network (CNN) architectures. However, Gupta et. al. [5] found that this approach does not make the most use of the geometric information encoded in the depth image.

In one of the earlier papers on the subject, Socher et. al. [6] proposed a CNN-RNN architecture in which CNNs extract low-level translation-invariant features for RGB and depth images independently, then RNNs generate high-level global features. Eitel et. al. [7] used a similar

approach in which two different CNNs process the RGB and depth images independently, then fuse the output after passing it through two fully connected layers.

1.3 Problem Definition

In this paper, we aim to build an RGB-D deep learning object recognition model which makes good use of the depth information and outperforms RGB-only models. As input, the model takes an RGB-Depth image pair, and outputs a prediction of the class of the object present in the image. Through this process, we aim to develop an understanding of 3-dimensional images in the context of deep learning and gain insights that pave the way for further improvement in future research.

2. METHODOLOGY

2.1 Dataset

We used the Washington University RGB-D dataset containing images of 300 common household objects organized into 51 categories. The dataset was collected using a Kinect-style 3D camera. The dataset is 84 GB large and contains 207,920 RGB-D images [8].

2.2 HHA Geocentric Encoding of Depth Information

In 2014, Gupta et. al [5] proposed a geocentric embedding of depth images which transforms single-channel depth images into a 3-channel representation. This representation encodes horizontal disparity, height above ground, and angle with gravity for each pixel (referred to as the HHA embedding). In their paper, they demonstrated that extraction of features from HHA images using CNNs learned stronger representations and achieved higher performance than raw depth images.

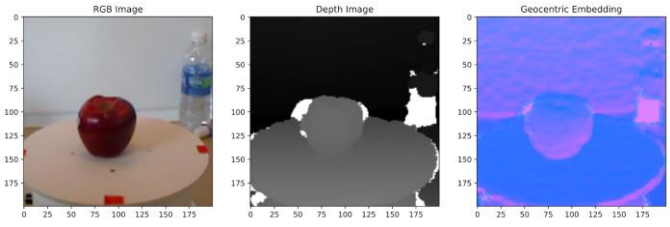


Figure 1: A sample image from the dataset

2.3 Proposed Architecture

We proposed and validated a deep learning architecture based on two independent CNN processing streams for the RGB and depth images, respectively. In the RGB processing stream, we make use of the proven ResNet50 2D CNN model to generate an RGB feature vector [9]. The depth image is transformed to the HHA geocentric encoding, then passed through a CNN feature extractor that we built from scratch. The RGB and depth feature vectors are then fused and passed through two fully connected layers before generating a class prediction.

Figure 2: The architecture of RGB/Raw depth model

2.4 Evaluation

We trained four different models using different combinations of RGB, raw depth, and HHA to isolate for the impact of each. Namely, we built and trained depth-only, HHA-only, RGB-Depth, and RGB-HHA models on our dataset.

We split the dataset into a training and test set by dedicating one object instance to the test set and leaving the rest for training. This was done to prevent data leakage since adjacent frames of the same object instance will look very similar.

3. RESULTS AND DISCUSSION

First, our results demonstrate that a deep learning model can achieve reasonable accuracy in recognizing objects in depth-only images. Our raw depth model achieved a test accuracy of 40.5%. Thus, depth images have utility beyond just providing auxiliary information to RGB images.

Table 1: Model performances

Model	Accuracy
Depth-Only	40.5%
HHA-only	48.0%
RGB w/ Raw Depth	54.7%
RGB w/ HHA	70.1%

Second, our results confirm that the HHA geocentric representation of depth images improves performance in deep learning object recognition models. The HHA-only model achieved notably higher accuracy than the raw depth model at 48.0%. The effect is even larger in the combined RGB and depth models. While RGB with raw depth achieved 54.7% accuracy, RGB with HHA achieved 70.1%. Thus, representing depth information in the HHA embedding improves performance for this model architecture.

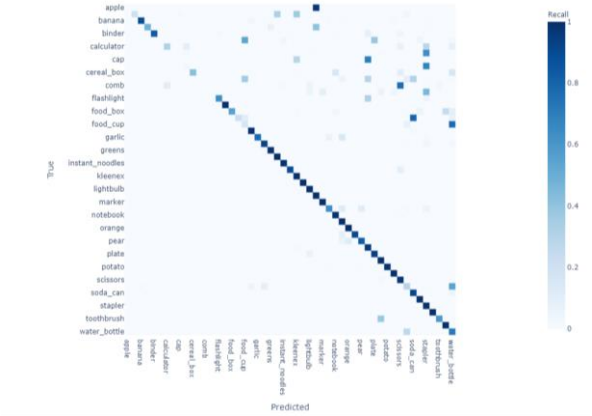


Figure 3: Confusion matrix of the RGB-HHA model

Finally, we examined the learned weights of the RGB-Depth models to further assess how useful the depth information was to the models. We did this by looking at the weights of the dense layer following the concatenated feature vector, and compared the weights associated with the RGB feature vector and the depth feature vector. We found the magnitudes of weights associated with depth information to be comparable to that of RGB information, especially when encoded in HHA.

Table 2: Statistics of learned weights

Statistic	*RGB w/ Raw Depth*		*RGB w/ HHA*	
	RGB	Raw Depth	RGB	HHA
Mean Abs. Value	0.0449	0.0243	0.0233	0.0234
Max	0.8412	0.1694	0.0467	0.0467
Min	-0.9376	-0.1694	-0.0467	-0.0467
Std. Dev.	0.0712	0.0286	0.0269	0.0270

4. CONCLUSIONS AND FUTURE WORK

With the utility of depth information in object recognition models demonstrated, future research may be directed at optimizing performance further and pursuing the maximum possible accuracy. Our architecture makes use of two independent processing streams for the RGB and depth images; perhaps an architecture where information from the opposite stream is allowed to gradually seep in before the final fusion would result in a model that is more aware of the "full picture." Another promising application of depth information in computer vision is object segmentation, where 3-dimensional structure is especially important. This would be an interesting avenue for further research.

5. REFERENCES

[1] S. Zia, B. Yuksel, D. Yuret and Y. Yemez, "RGB-D Object Recognition Using Deep Convolutional Neural Networks," *IEEE International Conference on Computer Vision Workshops (ICCVW),* 2017.

[2] D. Griffiths and J. Boehm, "A Review on Deep Learning Techniques for 3D Sensed Data Classification," *Remote Sensing,* vol. 11, no. 12, p. 1499, 2019.

[3] Y. Gao, F. Sohel, M. Bennamoun, M. Lu and J. Wan, "Rotational Projection Statistics for 3D Local Surface Description and Object Recognition," *International Journal of Computer Vision,* vol. 105, no. 1, pp. 63-86, 2013.

[4] Y. LeCun, "Deep learning & convolutional networks," *IEEE Hot Chips 27 Symposium (HCS),* 2015.

[5] S. Gupta, R. Girshick, P. Arbeláez and J. Malik, "Learning Rich Features from RGB-D Images for Object Detection and Segmentation," *Computer Vision – ECCV 2014,* pp. 345-360, 2014.

[6] R. Socher, B. Huval, B. Bhat, C. D. Manning and A. Y. Ng, "Convolutional-Recursive Deep Learning for 3D Object Classification," *Proceedings of the 25th International Conference on Neural Information Processing Systems,* 2012.

[7] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2015.

[8] K. Lai, L. Bo, X. Ren and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," *IEEE International Conference on Robotics and Automation,* 2011.

[9] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly and N. Houlsby, "Big Transfer (BiT): General Visual Representation Learning," 2020.

Our work can be found at:
https://github.com/Awni00/3d-object-classification

Video Summarization Tool

T. J. Hu[1], Bhavini Rathod[2], Nick Pysklywec[3], Mihir Kadiya[4], Andy Huang[5]

Ad Astra, Western AI
Western University, 1151 Richmond St. London ON, N6A 3K7 Canada.

1 e-mail: junruihu@gmail.com
2 e-mail: brathod@uwo.ca
3 e-mail: npysklyw@uwo.ca
4 e-mail: mkadiya@uwo.ca
5 e-mail: andy0207huang@gmail.ca

Abstract: *The ultimate goal of this work is to develop a video summarization search engine that allows users to identify an object in a video stream in a way that is analogous to finding a word in a text document. Detectron2, a library for real time object detection and segmentation, is used to sample videos to extract what objects appear in individual frames. This metadata is saved as high-dimensional vector embeddings using BERT. Topic modeling is then run on the associated subtitles to automatically infer video-chapters to further aid semantic search. Our model shows promising 80% detection confidence after training on the initial annotated images. The model can be implemented in research across multiple disciplines to increase efficiency when analyzing long video footages. Future work will enable our model to learn online so that users can easily label novel objects and activities for the model to learn.*

1. INTRODUCTION

1.1 Motivation

We are currently in an information age, where vast amounts of data continue being produced. A lot of that information is in the form of video data. According to a recent Cisco study, by 2021, 82% of consumer internet traffic will be video [1]. This is not a surprise as a lot of the video traffic that we consume from popular video apps like TikTok to large video search engines like YouTube are omnipresent in today's digital age.

Video data has the unique property that it has to be viewed in order for visual information to be extracted from it. For humans to view thousands of hours of video data can be time consuming to say the least. Our team is looking to find a way to automate the process of finding information in videos. We have built an automated search tool that summarizes video content. This work can be useful for a lot of applications to objects in a video to find it's location (timestamp).

A video search tool that can summarize the video feeds of "lengthy-videos" to save time and promote efficiency.

The motivation of creating this web application was to implement a solution to a problem that can combine. Our current application uses analysis of image segmentation to suggest video locations to where the object of interest is, and allows for search & highlighting of appearances within the video

1.2 Related Works

Examination of current literature was done, and there existed many different approaches to be taken when considering video summarization. In one instance, researchers converted videos and transcripts to a semantic space, and clustering techniques were applied to this data [2]. This method was able to get a 58% average score relative to manual video summaries.

In another case, an algorithm was developed to analyze the difference between two frames of a video, and by doing this determine “key frames” of a video [3].

1.3 Problem Definition

We are trying to address the problem of video search tools that can identify an object of interest in a video, and searching video data in an automated fashion. This requires object detection analysis and being able to highlight or segment an object from an image frame in the video. In our application, we touched on creating customized objects and trained the data Detectron’s Machine learning model.

2. METHODOLOGY

To summarize videos object recognition was first considered. Rather than develop a model for this, the PyTorch-based modular detection library Detectron2 (**Facebook Research**)[4] was used. Detectron2 provides capability for object detection in an image, furthermore segmentation. The idea was that this becomes useful for determining the given objects in an image.

To make use of Detectron2, we divided videos into frames of a frame rate at 1 fps (1 frame cut per second of video). Each frame was then sent to Detectron2, and the various objects in that frame were recorded along with this current frame number. After analyzing each frame of a video, this recorded frame and object data was sent to the frontend UI interface as a JSON and parsed to be shown as timestamps to the user.

This remains the basic pipeline for operation, furthermore, Detectron2 also provides functionality for adding custom objects to it. Custom objects were added to Detectron2 for a greater expansion of domain specific objects. Images used to train custom objects were scraped from google images using a bulk image downloader chrome extension. For 50 of scraped images annotated to Detectron2 for training, the model yielded **80% accuracy** for testing object recognition for the custom item.

Figure 1: A custom image being identified by Detectron2

3. RESULTS AND DISCUSSION

We would like to validate our work by comparing our application’s detection mechanism to humans who will mark the locations of objects and record their timestamps as our gold standard marker.

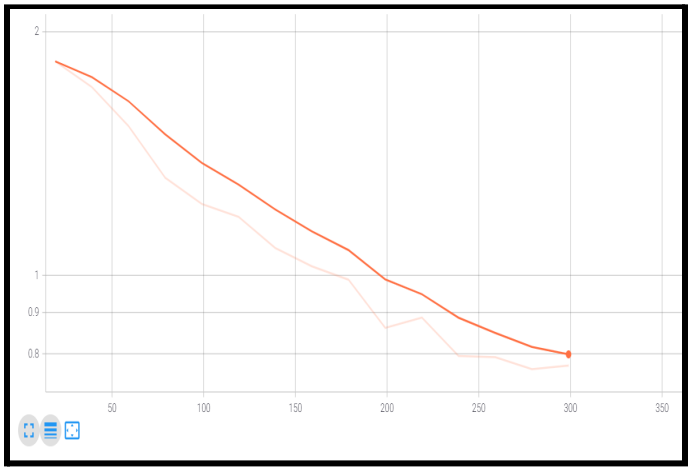


Figure 2: Total loss iterated over 300 epochs

The plotted graph displays the total loss of the model for a custom object in our model over a series of epochs. The Detectron2 model did very well at adding custom objects to itself, even with a "small" amount of train images (50). These results indicate that training of Detectron2 onto more images will generate a more accurate object classifier. Some uncertainty appeared however with some objects when placement in a given was not two dimensional.

For instance, some objects appeared facing into the photo, but since many google images for training were facing horizontal or vertical, Detectron2 had issues classifying these objects sometimes. Another issue was with misclassification, as some objects were being misclassified as a vegetable, but were a vehicle. The given vehicles were boats, and the vegetable was broccoli, so the issue likely is associated with the polygon shape that Detectron2 uses for recognition.

4. CONCLUSIONS AND FUTURE WORK

Overall, the attempted solution aims to solve the problem of detecting objects from a video using NLP topical analysis provided by Detectron2. An automatic video search tool such as this has a variety of applications in many different industries such as surveillance, custom object detection and many more.

In a given video file, there could be a series of frames that detects a certain object, however, some frames in the range can be blurry or distorted which Detectron2 does not recognise and therefore the algorithm splits the object into two, even though they should be one object. One of the ways to address this issue would be to create a smooth function that can predict the gaps in a series of frames which are blurry or distorted.

The method of data collection for new object addition is cumbersome, so another priority that remains is to automate this. This can be done by exploring use of the *Toronto Data Platform* which assists human annotation, or by developing a tool. Additionally, fixing the issues pertaining image orientation in 2D space can be fixed using various data augmentation techniques.

REFERENCES

[1] ADG Creative (2020) Available: https://www.adgcreative.net/resources/by-2021-82-of-consumer-internet-traffic-will-be-video-are-we-ready/

[2] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkilä, N. Yokoya, "Video Summarization using Deep Semantic Features, " *Lecture Notes in Computer Science* , pp. 361-377 (2017). [Online]. Available: https://arxiv.org/abs/1609.08758

[3] C. Gianluigi, S. Raimondo, "An innovative algorithm for key frame extraction in video summarization," *Journal of Real-Time Image Processing,* vol. 1, no. 1, pp. 69 - 88 (2006). [Online]. Available: https://doi.org/10.1007/s11554-006-0001-1

[4] Y. Wu, A. Kirillov, F. Massa, W.Y Lo, R. Girshick, Detectron2, (2019). Available: https://github.com/facebookresearch/detectron2

Voiceprint Identification

Alastair Noble, Harley Latsky, Eli James, Alex Beamish

QMIND – Queen's AI Hub
Queen's University, Kingston, Ontario K7L 3N6, Canada. Queen's

1-email alastair.noble@queensu.ca
2- email harley.latsky@queensu.ca
3-email 19ewj@queensu.ca
4-email 19akb3@queensu.ca

Abstract: *Voiceprint recognition is the process by which a trained model identifies the specific individual who is speaking arbitrary phrases. Speaker recognition is distinct from speech recognition, which involves converting spoken words to text, but these two types of systems are often combined in practice, as is the case with virtual assistants such as Siri, Alexa and Google Assistant. Currently, largely due to the training approaches used, many speaker recognition models can only identify speakers when a specific phrase is uttered by the speaker. In an attempt to address this issue, and to create a complete voiceprint recognition system, we have developed a multi-faceted software application in python. The application includes a neural network, a set of features that allow for flexible training on multiple words and phrases, and a user-friendly GUI. Our application can identify speakers with roughly 80% accuracy, although we are currently working to improve that. We plan on continuing to experiment with different approaches for training, with the goal of maximizing identification accuracy even when the speaker uses novel words and phrases. Future uses of our software could include audio captioning/transcription, and authentication for security or customer service purposes.*

1. INTRODUCTION

1.1 Motivation

Accurate voiceprint recognition systems are becoming increasingly important to develop more accessible and secure technology [1]. Voiceprint recognition models are used in areas such as voice assistants, two-step authentication security systems and customer service [2]. Two areas in which existing voiceprint recognition systems could improve are their ability to accurately differentiate between two speakers and their ability to accurately identify whether it is the same person speaking but under different conditions. These systems identify more than just speech itself; they are also taking background noise, echoes, and other sound features into account, all while constrained by the hardware upon which the sound source is recorded.

We sought to build a voiceprint recognition model which could accurately differentiate between speakers in real time.

1.2 Related Works

In the past, the most common approaches to voiceprint recognition and verification models were Gaussian Mixture Models (GMMs) and hidden Markov Models (HMMs). The more common approach currently, and the one we developed, was using a convolutional neural network (CNN). The advantages and disadvantages of these approaches can be summarized by Mingyu Ma's paper [3], or by this Microsoft Research Paper [4] from 2014. One advantage of using a CNN is that it promotes more flexibility in handling the natural variability in speech. For a few years, the CNN has been known to be an effective approach for identifying and verifying voice using machine learning.

1.3 Problem Definition

Using a CNN, we sought to build a model which could accurately identify speakers in real time using multi-phrase voiceprint recognition with the aim of discovering improved ways of collecting and cleaning

training data to maximize the accuracy of identification.

2. METHODOLOGY

2.1 Prototype

In building a multi-phrase voiceprint recognition system, our team started by implementing a single-phrase system. This initial system was modelled after Jurgen Arias's work on voice classification, and used Librosa, Keras (sequential neural network), and MFCCs (Arias, 2019). To fully test our initial single voice classification system, we used data from an open-source Alexa dataset containing 86 users saying the word Alexa 4 times. Our group added data of ourselves saying Alexa 4 times to add to this data and ensure that the model is working through testing.

2.2 Data Processing

To convert audio to data that can be understood by the model, our group split the original audio file into 40 different audio chunks. For each chunk, a coefficient, namely a Mel-Frequency Cepstrum Coefficient (MFCC) was created. This MFCC was generated based on the frequency of the audio chunk put through a Fourier transform to bias the coefficient towards small changes in frequency. This created a 40 long decimal array that was fed into the model.

Before the training of the model, the data needed to be processed in order to avoid overfitting and ensure that the model focused on the correct indicators. An example of a situation that we wanted the model to avoid paying attention to is the time in-between the start of the recording and the first sound. To prevent the model from focusing on this, we utilized Librosa to add a hamming window to our data, which softened out large changes in audio. We also used a noise removal formula to help limit the bias of background noise from different microphones. Using the single phrase system, our team tried to increase the efficiency of the model and create code that can be expanded upon with future iterations of the model.

2.3 Final Design

After obtaining satisfactory results with the single phrase system, our team split up to build out as many features as possible for the final design. A data collection system that works with the previous model requires sentences to be split up into words. Our team used PyDub (the python library) to analyze full sentences and return separate audio files of all the individual words used. This relied on the silences in-between words and greatly sped up the data collection process. Next, we increased the accuracy of the model by iterating on the model. For example, the audio chunk number of 24 was determined a better indicator than the original 40. The last task was to bring all these pieces together into an application with a simple UI. The most significant feature of the UI is to test the model live using live input from a microphone while continually updating the UI.

3. RESULTS AND DISCUSSION

We trained the final model on 4 subjects using 35 seconds of speech data per subject. We broke the data up into between 60 and 70 audio chunks based on how fast the subject spoke. We evaluated the model in 2 ways: using prerecorded test data of each subject, and with live input during a meeting between the 4 subjects. The model had 88% accuracy on the test data. It averaged 80% accuracy during the live input, however it varied with different conversations. During presentations, when subjects spoke one at a time without interruption, the model reached 88% accuracy, however during regular meetings where there was a lot of back and forth, the accuracy averaged 72%. The lower accuracy during the live input was expected for a few reasons. First, conversations between subjects often have audio input from more than one party, in the form of background noise or talking over each other. And second, we had to collect the audio over set intervals to update the GUI, so we had to balance the speed of the prediction in the GUI with the quality of the audio chunks we collected.

We chose to only use 35 seconds of audio data for training because we prioritized usability during

meetings and presentations, and we wanted to be able to easily add subjects to the training set during the demonstration at CUCAI 2021. For different applications with more subjects or where higher accuracy is required, more training data should be used so that the model has more data to learn the unique identifiers of each subject.

A limitation we had was that we were displaying the predictions live. This meant we had to cut audio into intervals before breaking it up naturally, so we often had words cut in half at the beginning and end of intervals. For different applications, for example transcription, the audio would be broken up more naturally after it was all recorded, and would have better accuracy.

4. CONCLUSIONS AND FUTURE WORK

The progression from a single phrase model to a multi-phrase model went smoothly, and the final model performed well considering the limited amount of speech data that was used to train it. In the final stages of development, various software components were successfully merged into a usable application. As outlined in the previous section, there are certain limitations on what the final model can do, and more robust testing under various conditions is still required. Nonetheless, results so far have been very promising, and there do not appear to be any insurmountable barriers to further improvement of the model. Future work could include continuing to search for better phrases to train the model on, and continuing to optimize the identification accuracy through tuning of the model parameters.

Many of the major applications of voice identification technology are in authentication for security and customer service purposes, and in audio captioning. In the security and customer service space, our application could be used in conjunction with other methods to allow for real-time, continuous verification of identity while a speaker is speaking. Our application could also be used in digital audio captioning to identify a speaker across multiple audio files. Ultimately, speaker identification is a very general problem, so our application could be useful in many different contexts.

REFERENCES

[1] S.M. Schwartz (2017). Multi-Agent Path Planning for Lo-cating a Radiating Source in an Unknown Environment.Master's Thesis, Department of Mechanical Engineer-ing, Embry-Riddle Aeronautical University.

[2] M. Vrigkas, C, Nikou, and I. Kakadiaris, "A Review of HumanActivity Recognition Methods", Front. Robot. AI, 16 November 2015.

[3] M. Alzahrani, S. Kammoun, "Human Activity Recognition: Chal-lenges and Process Stages",International Journal of Innovative Re-search in Computer and Communication Engineering, Vol. 4, Issue 5,May 2016.

[4] C. McDaniel, S. Quinn, "Developing a Start-to-Finish Pipeline forAccelerometer-Based Activity Recognition Using Long Short-TermMemory Recurrent Neural Networks",PROC. OF THE 17th PYTHONIN SCIENCE CONF, 2018

[5] J. Arias, "Voice Classification," *GitHub*, 07-Dec-2019. [Online]. Available: https://github.com/jurgenarias/Portfolio/tree/master/Voice%20Classification. [Accessed: 17-Mar-2021].