CIND 860 Capstone Project

Analysis of Sign Language MNIST using Deep Learning

By: Muhammad Zaka Shaheryar (500648718)

Ryerson University

Agenda

- **Introduction**
- **Data Description**
- **Research Question**
- **Literature Review**
- **Data Preparation**
- **Experimental Design**
- **Predictive Modelling**
- **Evaluation**
- **Hyperparameter Tuning**
- **Analysis Limitations**
- **Conclusion**

Introduction

- **Chosen Dataset:** Sign Language MNIST (Kaggle)
- **Theme:** Image Classification using Machine learning and Deep learning
- **Project Objective:** Examining and Comparing Convolutional Neural Networks (CNNs) with traditional classifiers like KNN, NB, SVM, RF and XGBoost
- **Methodology and Tools:** Python libraries for deep learning and traditional machine learning algorithms, and employing a systematic data analysis process approach.
- **Project Approach:** Initial analysis, Exploratory analysis (EDA), Data preparation, Experimental design, Model building, Performance evaluation, and Conclusion.
- **Importance of Study:** Sign language is crucial for effective communication for the 300 million deaf and 1 million mute individuals globally, addressing the challenges of hearing impairment caused by various factors.

Dataset Description

- Two CSV files: sign_mnist_train and sign_mnist_test
- Each row represents an image, totaling 34627 hand gesture images.
- 27455 training and 7172 testing grayscale images
- Total columns are 785 including target column “label”
- “label” is distributed over 24 ASL classes or alphabets (excluding J and Z)
- Pixel intensity ranging from 0-255
- All variables (label and 784 pixels are numeric)

Research Questions

No.	Research Question
1	How do accuracy, computational efficiency, and stability of CNN compare to traditional machine learning techniques such as K-NN, Naïve Bayes, Random Forest, SVM, or XGBoost in classifying ASL images?
2	How to optimize the CNN model by hyperparameter tuning and better architecture that exhibits the greatest efficiency in classifying ASL images considering accuracy and computational resources?
3	When compared to Naive Bayes (NB), how effective is GNB at identifying the true cause of the categorial target label of English alphabets in the context of the Sign language MNIST dataset?

Ryerson University

Literature Review

- My analysis is replication of (Karayaneva et al) analysis
- I used similar models: KNN, SVM, RFC, and CNN as they did
- I got almost similar results to theirs.
- Their ranking of model according to accuracy is Neural network, then KNN, and 3rd is SVM.
- I got KNN as first, SVM as 2nd and RF as third as shown in the overall evaluation slide

Data Preparation- Feature Selection

- Filter approach is used first to find the top 25% of 784 pixels using the chi-square test which evaluates the association between each pixel and ASL labels and selects the top-performing labels based on the highest score.
- This reduces the column size of train data to 195 features. Then top 25% of 195 is found using a hybrid approach which uses recursive feature elimination (RFE) to iteratively build a model to identify a subset of pixels that contributed most to model accuracy.
- The model used is a Random Forest classifier to find the best pixels related to the target. Therefore, the final features in train data have been reduced to 48 as shown in next slide.

Data Preparation - Feature Selection

	Features	Importance
1	pixel354	0.003464
2	pixel520	0.003418
3	pixel211	0.003319
4	pixel239	0.003303
5	pixel246	0.003300
6	pixel413	0.003277
7	pixel355	0.003240
8	pixel266	0.003197
9	pixel238	0.003179
10	pixel274	0.003138
11	pixel321	0.003137
12	pixel441	0.003133
13	pixel493	0.003003
14	pixel273	0.002956
15	pixel237	0.002880
16	pixel576	0.002837
17	pixel265	0.002814
18	pixel382	0.002796
19	pixel302	0.002778
20	pixel385	0.002768
21	pixel245	0.002762
22	pixel159	0.002709
23	pixel376	0.002702
24	pixel519	0.002699
25	pixel327	0.002686
26	pixel352	0.002653
27	pixel215	0.002635
28	pixel353	0.002625
29	pixel350	0.002611
30	pixel772	0.002582
31	pixel465	0.002574
32	pixel187	0.002568
33	pixel437	0.002553
34	pixel240	0.002553
35	pixel217	0.002548
36	pixel381	0.002542
37	pixel466	0.002540
38	pixel377	0.002533
39	pixel469	0.002524
40	pixel210	0.002512
41	pixel411	0.002505
42	pixel440	0.002501
43	pixel348	0.002479
44	pixel605	0.002475
45	pixel387	0.002464
46	pixel326	0.002451
47	pixel438	0.002451
48	pixel468	0.002450

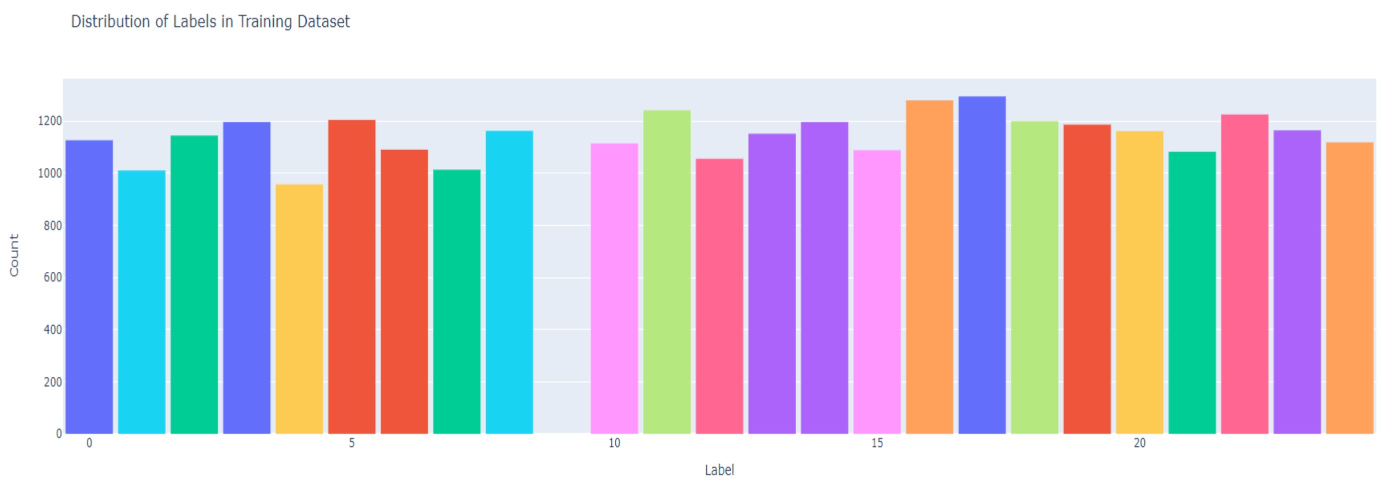
Data Preparation - Feature Selection

```
selected_features = ['pixel354', 'pixel520', 'pixel211', 'pixel239',
'pixel246',
                     'pixel413', 'pixel355', 'pixel266', 'pixel238',
'pixel274',
                     'pixel321', 'pixel441', 'pixel493', 'pixel273',
'pixel237',
                     'pixel576', 'pixel265', 'pixel382', 'pixel302',
'pixel385',
                     'pixel245', 'pixel159', 'pixel376', 'pixel519',
'pixel327',
                     'pixel352', 'pixel215', 'pixel353', 'pixel350',
'pixel772',
                     'pixel465', 'pixel187', 'pixel437', 'pixel240',
'pixel217',
                     'pixel381', 'pixel466', 'pixel377', 'pixel469',
'pixel210',
                     'pixel411', 'pixel440', 'pixel348', 'pixel605',
'pixel387',
                     'pixel326', 'pixel438', 'pixel468']
```

Data Preparation - Pandas Profiling Report

- Comprehensive analysis of a train dataset with 27455 observations and 49 variables.
- 0% missing data
- 0% duplicate rows
- High correlations among certain pixel values, indicating redundancy or strong dependencies between features.
- Ignorable outliers and zeros (extreme lightness in pixel).
- All 48 selected features have Gaussian-like or almost normal distribution.
- label column which is the target column shows uniform distribution which also means almost no class imbalance as shown in the next slide.

Data Preparation
Class Distribution



Data Preparation - Label Images

Ryerson University

Experimental Design

- To avoid data leakage only a train dataset is used to train the model
- The model is tested on new data, which is test data already in a separate file, so no need to split the dataset into training and testing data.
- Six models are used in this project, namely K-Nearest Neighbours, Naïve Bayes, Support Vector Machine, Random Forest, XGBoost, and Convolutional Neural Network.
- KNN is taken as Baseline model to be used as reference for other models. Other models' performance is checked compared to KNN.
- To check the stability of most model K fold cross validation is used and K is taken as 10.
- Therefore, mean validation accuracy and variance is reported together with test accuracy.

Predictive Modelling- KNN Baseline

Classification Report (KNN):

	precision	recall	f1-score	support
0	0.66	0.98	0.79	331
1	0.97	0.93	0.95	432
2	0.98	0.93	0.96	310
3	0.79	1.00	0.88	245
4	0.91	0.94	0.93	498
5	0.86	0.84	0.85	247
6	0.60	0.74	0.66	348
7	0.77	0.70	0.74	436
8	0.51	0.44	0.47	288
10	0.60	0.44	0.51	331
11	0.64	0.99	0.78	209
12	0.80	0.58	0.67	394
13	0.77	0.26	0.39	291
14	0.99	0.70	0.82	246
15	0.80	0.79	0.79	347
16	0.48	0.87	0.61	164
17	0.31	0.49	0.38	144
18	0.64	0.69	0.66	246
19	0.41	0.56	0.47	248
20	0.55	0.62	0.58	266
21	0.63	0.56	0.60	346
22	0.70	0.64	0.67	206
23	0.76	0.55	0.64	267
24	0.60	0.46	0.52	332
accuracy			0.70	7172
macro avg	0.70	0.70	0.68	7172
weighted avg	0.72	0.70	0.70	7172

Predictive Modelling- Naïve Bayes

Classification Report (NB):

	precision	recall	f1-score	support
0	0.50	0.72	0.59	331
1	0.71	0.91	0.80	432
2	0.71	0.68	0.70	310
3	0.45	0.22	0.30	245
4	0.64	0.76	0.70	498
5	0.35	0.44	0.39	247
6	0.32	0.40	0.36	348
7	0.43	0.33	0.37	436
8	0.17	0.22	0.19	288
10	0.25	0.23	0.24	331
11	0.27	0.24	0.25	209
12	0.53	0.28	0.36	394
13	0.12	0.09	0.10	291
14	0.44	0.36	0.39	246
15	0.33	0.21	0.25	347
16	0.23	0.62	0.34	164
17	0.05	0.06	0.06	144
18	0.20	0.13	0.16	246
19	0.30	0.35	0.32	248
20	0.25	0.13	0.17	266
21	0.38	0.26	0.31	346
22	0.27	0.50	0.35	206
23	0.36	0.53	0.43	267
24	0.38	0.16	0.23	332
accuracy			0.39	7172
macro avg	0.36	0.37	0.35	7172
weighted avg	0.39	0.39	0.38	7172

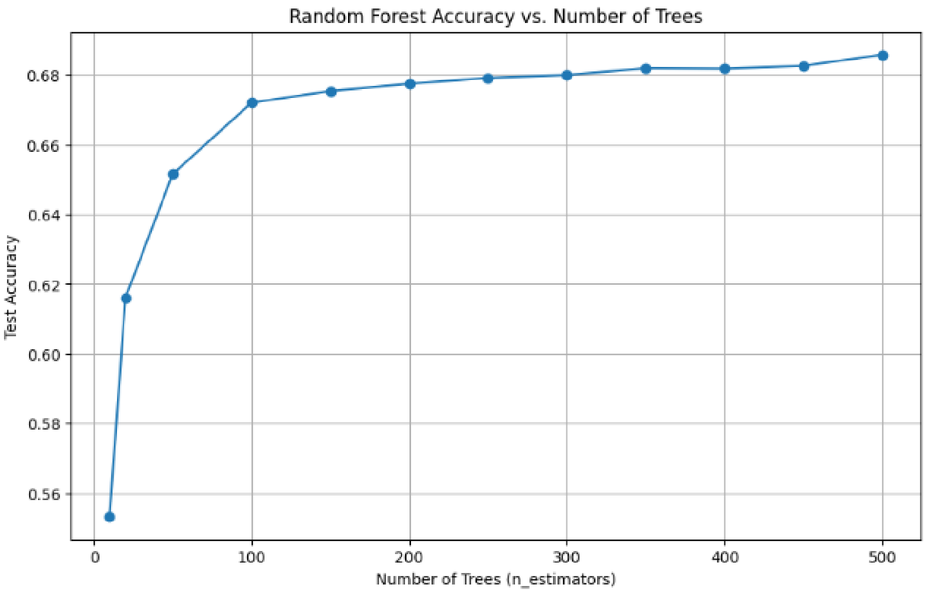
Ryerson University

Predictive Modelling- SVM

Classification Report (SVM):

	precision	recall	f1-score	support
0	0.83	0.96	0.89	331
1	0.94	0.94	0.94	432
2	1.00	0.98	0.99	310
3	0.66	0.93	0.77	245
4	0.90	0.90	0.90	498
5	0.73	0.81	0.77	247
6	0.60	0.67	0.64	348
7	0.82	0.72	0.77	436
8	0.44	0.51	0.47	288
10	0.58	0.48	0.52	331
11	0.75	0.99	0.85	209
12	0.77	0.64	0.70	394
13	0.50	0.35	0.41	291
14	1.00	0.78	0.88	246
15	0.78	0.64	0.70	347
16	0.56	0.87	0.68	164
17	0.22	0.49	0.31	144
18	0.60	0.67	0.63	246
19	0.67	0.56	0.61	248
20	0.59	0.47	0.52	266
21	0.65	0.42	0.51	346
22	0.49	0.69	0.57	206
23	0.85	0.60	0.70	267
24	0.52	0.48	0.50	332
accuracy			0.69	7172
macro avg	0.69	0.69	0.68	7172
weighted avg	0.71	0.69	0.70	7172

Predictive Modelling- Random Forest Graph



Predictive Modelling- Random Forest

Random Forest Classification Report:

	precision	recall	f1-score	support
0	0.80	0.96	0.88	331
1	0.87	0.95	0.91	432
2	0.99	0.97	0.98	310
3	0.76	0.95	0.84	245
4	0.89	0.94	0.91	498
5	0.75	0.85	0.80	247
6	0.62	0.66	0.64	348
7	0.87	0.76	0.81	436
8	0.46	0.47	0.47	288
10	0.50	0.43	0.46	331
11	0.64	0.98	0.78	209
12	0.83	0.58	0.68	394
13	0.56	0.25	0.35	291
14	0.92	0.69	0.79	246
15	0.76	0.68	0.72	347
16	0.52	0.87	0.65	164
17	0.21	0.43	0.28	144
18	0.54	0.61	0.58	246
19	0.43	0.55	0.49	248
20	0.55	0.50	0.52	266
21	0.62	0.46	0.53	346
22	0.62	0.61	0.62	206
23	0.67	0.59	0.63	267
24	0.50	0.36	0.42	332
accuracy			0.68	7172
macro avg	0.66	0.67	0.65	7172
weighted avg	0.69	0.68	0.68	7172

Ryerson University

Predictive Modelling- XGBoost

Classification Report (XGBoost):

	precision	recall	f1-score	support
0	0.75	0.92	0.82	331
1	0.89	0.89	0.89	432
2	0.86	0.83	0.84	310
3	0.69	0.80	0.74	245
4	0.86	0.88	0.87	498
5	0.68	0.72	0.70	247
6	0.64	0.61	0.62	348
7	0.87	0.75	0.81	436
8	0.47	0.51	0.49	288
9	0.55	0.43	0.49	331
10	0.75	0.98	0.85	209
11	0.80	0.47	0.59	394
12	0.53	0.32	0.39	291
13	0.80	0.59	0.68	246
14	0.62	0.60	0.61	347
15	0.52	0.83	0.64	164
16	0.17	0.41	0.24	144
17	0.48	0.57	0.52	246
18	0.40	0.50	0.44	248
19	0.55	0.52	0.53	266
20	0.62	0.45	0.52	346
21	0.43	0.58	0.50	206
22	0.73	0.59	0.65	267
23	0.46	0.44	0.45	332
accuracy			0.64	7172
macro avg	0.63	0.63	0.62	7172
weighted avg	0.66	0.64	0.64	7172

Predictive Modelling- Simple CNN

- First step in CNN model is preprocessing step which include reshaping of input image to 4D tensor of shape (width, height, number of channels, 1). The number of channels is taken as 1. Also, the last dimension is 1 because it is a pixel are in greyscale format.

- Therefore, X_train_reshape has shape of (27455, 48, 1, 1)

- Secondly input tensor, (x_train_reshape) is converted into float data type and normalize to have value between 0 and 1.

- Finally, since target column is numerical, so it is converted to categorical using one hot coding.

Predictive Modelling- Simple CNN

Configuration of simple CNN model

optimizer='rmsprop’,

epochs=25,

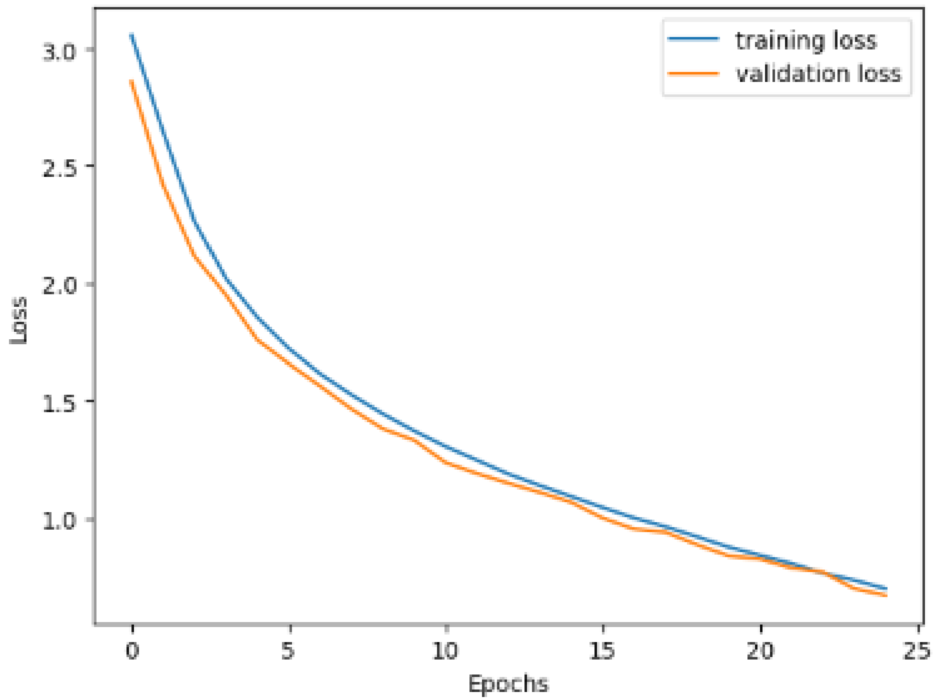
batch_size = 512

validation ratio=0.2

1 conv layers with 32 channels and (3x1 kernels) followed by a max pooling layer (2x1), and a final dense layer of 128 neurons before the output layer.

Ryerson University

Predictive Modelling- Simple CNN Loss Graph



Here, it seems underfitting as validation loss has a decreasing trend. Therefore, increasing the number of epochs might reduce validation loss further.

Ryerson University

Evaluation - Overall

Model	Test Accuracy (%)	Crosss validation variance	MCC	Train time (sec)	Test time (sec)	Peak memory (MiB)	Overall Rank
KNN Baseline	70	0.0000018					1st
NB	39	0.00015					6th
SVM	69	0.013	0.68	13.6	9.3		2nd
RF (300 tress)	68	0.014	0.67	23.4	0.5	12218	3rd
XGBoost	64	0.11	0.63	10.2	0.07	10192	4th
CNN	52						
CNN tuned	57						5th

Ryerson University

Evaluation - Best three Model

- KNN shows better accuracy (70%) compared to CNN (52%)
- KNN is the most stable model having the lowest cross-validation variance
- SVM is second best model which has an accuracy of 69 %
- SVM is third best stable model having cross validation variance of 0.013
- SVM has MCC of 0.68 which is the best among all models
- The training and test times for SVM are respectively 13.6 seconds and 9.3 seconds which is second best among models.
- Third best is Random Forest which has an accuracy of 68% and MCC of 0.67
- Random Forest (RF) has a cross-validation variance of 0.014
- RF train/test times are respectively 23.4 and 0.5 seconds
- RF has a peak memory consumption of 12218 MiB.

Evaluation – 4^{th} and 5^{th} ranked Model

XGBoost

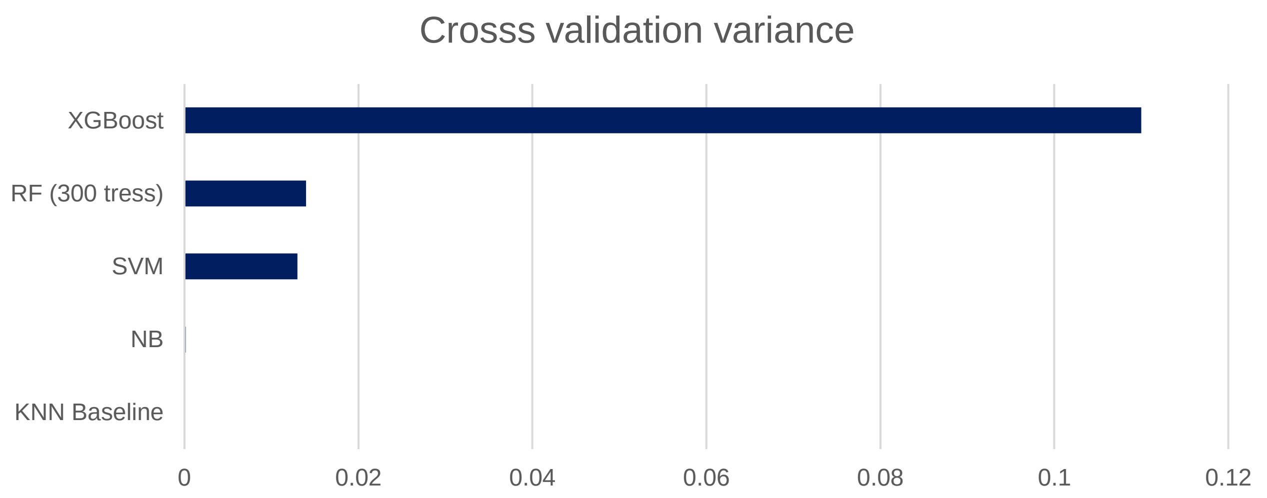
- The next best model (4^{th} rank) is XGBoost with having accuracy of 64 %
- It has MCC of 0.63 and it has cross validation variance of 0.11
- Furthermore, it is the top model concerning time efficiency with 10.2 seconds of training time and 0.07 seconds of testing time respectively.
- Moreover, it has peak memory consumption of 10192 MiB

CNN

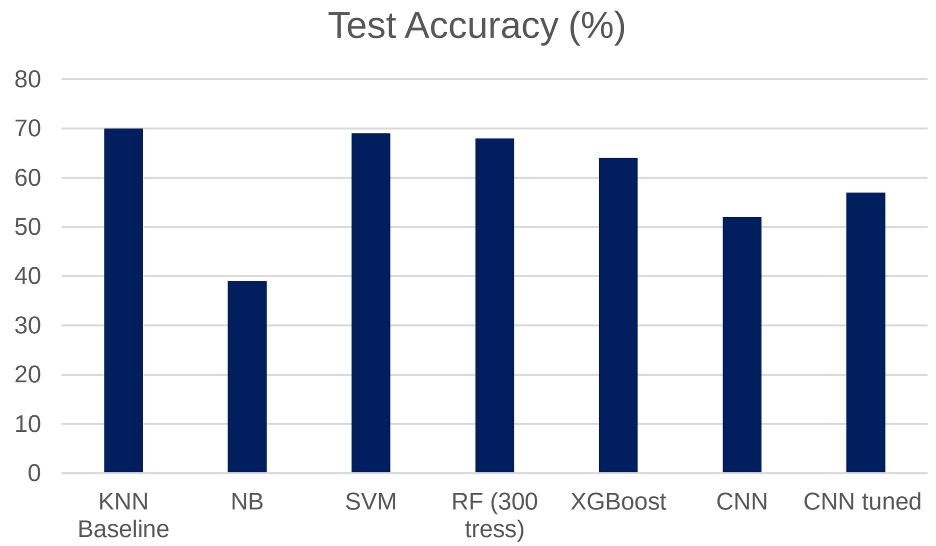
- It came out to be ranked 5^{th} which is contradiction with previous research
- Simple CNN has accuracy of 52%
- After hyperparameter tuning (increasing number of epoch), accuracy increased by 5% but still less than XGBoost.

Ryerson University

Evaluation - Stability

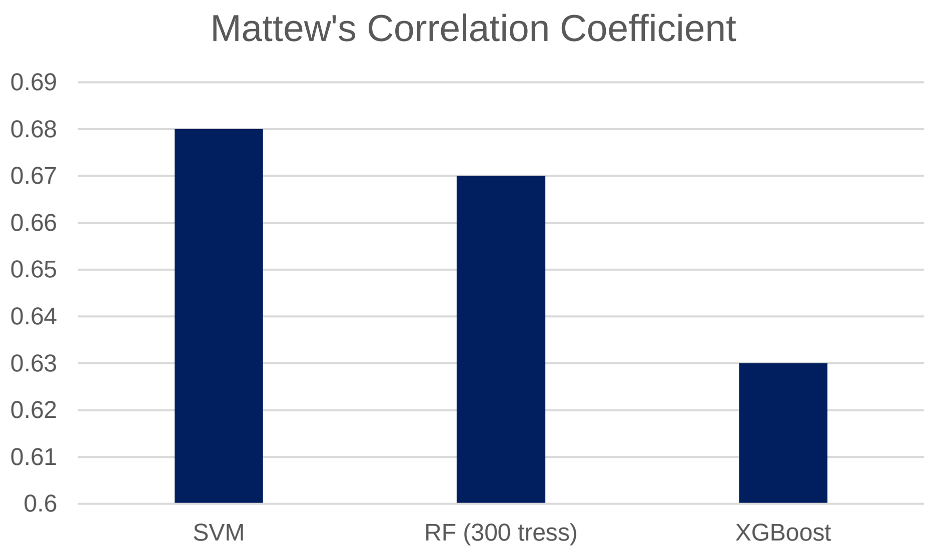


Evaluation -Effectiveness

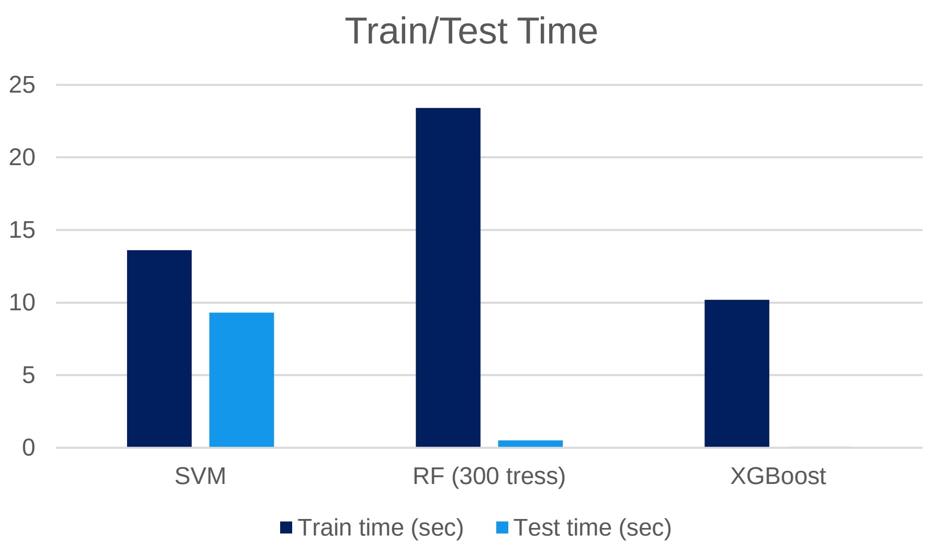


Ryerson University

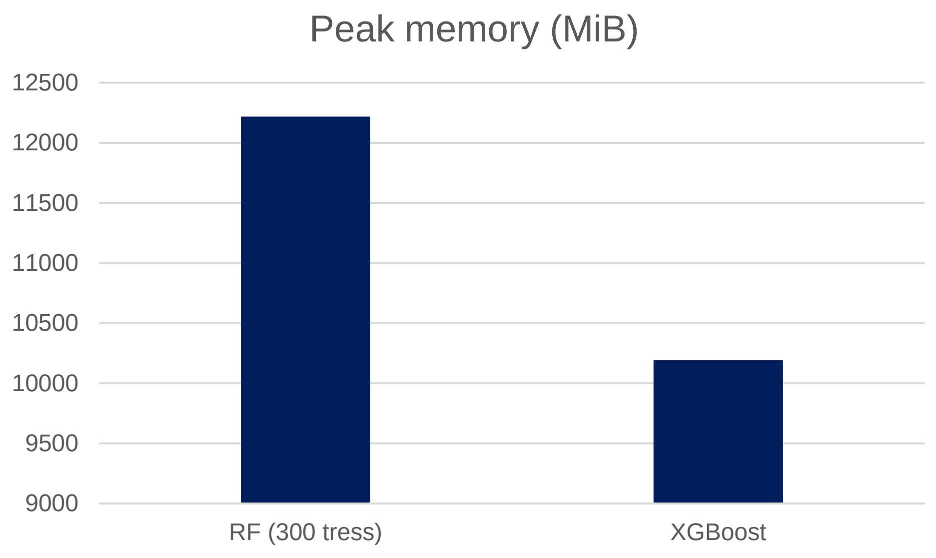
Evaluation –Effectiveness (MCC)



Evaluation – Time Efficiency

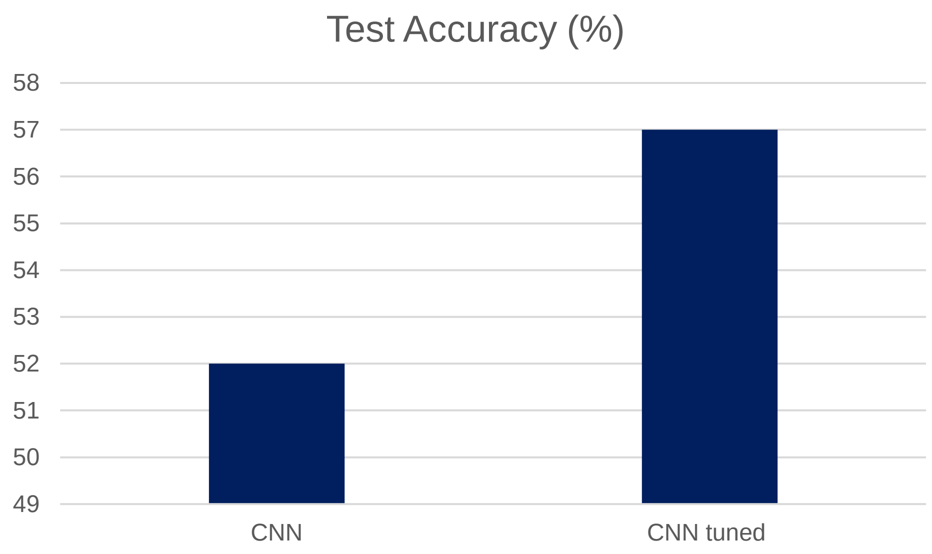


Evaluation - Memory Efficiency



Ryerson University

Hyperparameter Tuning



- The number of epochs that is chosen in CNN tuned model considering memory and time consumption is 50
- There is 5% increase in accuracy after tuning the model

Ryerson University

Analysis Limitations

- Different hyperparameter tuning for CNN model was also carried out but since the code is giving errors, therefore they have been left out of the analysis.
- Further investigation into CNN configuration and hyperparameter tuning is needed, as the accuracy is very low and it is unable to beat KNN baseline
- More data is needed, or data augmentation can help in better prediction.
- New features or combing features can be added that can help in increasing the efficiency of the model.
- Other variants of CNN can also be explored such as MobilNet, AlexNet, LeNet, VGG, etc.
- Research question 3 is excluded due to time constraints.

Conclusion

- Privacy of personal data should be kept.
- Three different categories of machine learning models have been used in the project
- The results and analysis limitation show that KNN is the best-performing model for this project with a test accuracy of 70%
- CNN doesn't perform well as discussed before which can be a topic to explore for future studies
- In particular grid search can be used available in the scikit-learn python library to tune the hyperparameters of Keras’s CNN model
- In addition to that Bayesian network can be explored to find the cause of labels in the sign language MNIST dataset

References

[1] Kumar et al (2023). Mediapipe and CNNs for Real-Time ASL Gesture Recognition. Retrieved from https://arxiv.org/abs/2305.05296

[2] Pigou et al (2015). Sign Language Recognition Using Convolutional Neural Networks. Retrieved from https://link-springer-com.ezproxy.lib.torontomu.ca/chapter/10.1007/978-3-319-16178-5_40

[3] Eman et al (2023). Arabic Sign Language Recognition using Convolutional Neural Network and MobileNet. Retrieved from https://journals-scholarsportal-info.ezproxy.lib.torontomu.ca/details/2193567x/v48i0002/2147_aslrucnnam.xml

[4] Lanxi et al (2022). American Sign Language Recognition Based on Machine learning and Neural Network. Retrieved from https://ieeexplore-ieee-org.ezproxy.lib.torontomu.ca/document/9943235

[5] Hasan et al. (2020). Classification of Sign Language Characters by Applying a Deep Convolutional Neural Network. Retrieved from https://ieeexplore-ieee-org.ezproxy.lib.torontomu.ca/document/9333456

References continued

[6] Karayaneva et al (2018). Object Recognition in Python and MNIST Dataset Modification and Recognition with Five Machine Learning Classifiers. Retrieved from https://www.joig.net/uploadfile/2018/0717/20180717055805469.pdf

[7] Khlifia et al.(2012). New approach using Bayesian Network to improve content based image classification systems. Retrieved from https://arxiv.org/abs/1204.1631

[8] OpenAI. (2023). ChatGPT (GPT-4 model) [Large language model]. https://chat.openai.com/chat

[9] Kaggle (2018). Sign Language MNIST. Retrieved from https://www.kaggle.com/datasets/datamunge/sign-language-mnist

[10] Jason. (2022). How to Grid Search Hyperparameters for Deep Learning Models in Python with Keras. Retrieved from https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

References continued

[11] Yadav et al. (2019). Recognition of Static Hand Gestures in Indian Sign Language.

[12] Singh, A. (2020, July). Sign language MNIST problem: American Sign Language. Retrieved from https://medium.com/@abhkmr30/sign-language-mnist-problem-american-sign-language-48896ea960e0

Ryerson University

Thankyou for Listening