CS4662 Project Final Report

Detecting Different Letters of the Alphabet in American Sign Language with Machine Learning

Table of Contents

Team Members

Leonard Garcia (300302643)

Role: Team Lead

Responsibilities:

- Facilitate communication between members
- Identify project goals, set deadlines, and ensure deadlines are met
- Initial data preparation

Karen Quan (401037725)

Role: Documentation Lead

Responsibilities:

- Cataloging project progress
- Creating data visualizations

Bryan Chan (306330198)

Role: Traditional Machine Learning Lead

Responsibilities:

- Research and implement traditional machine learning models
- Tune and improve performance of traditional machine learning models

Han Cao (400497471)

Role: Deep Learning Lead

Responsibilities:

- Research and implement Neural Networks
- Tune and improve performance of Neural Networks

1. Introduction

1.1 Background

American sign language (Asl) is a form of sign language developed in the United States commonly used by those who are deaf or hard of hearing. It is a complete language with similar linguistic properties as spoken languages, albeit with grammar that differs from English. The American sign language alphabet is a set of 26 letters that correspond to the 26 letters of the English alphabet. It is used for fingerspelling proper nouns.

1.2 Project Goal

The purpose of this project is to train a variety of machine learning models to determine which can most accurately distinguish between images of the different letters of the alphabet in American Sign Language. The importance of this project stems from its potential to enhance and expand the translation, communication, and education of society. This project can be used to translate between american sign language and spoken English, thus facilitating communication between users of both languages. It can also be used to educate anyone who would like to learn american sign language. This will increase accessibility for those who are deaf or hard of hearing.

1.3 Project Design Principles

Two main approaches were used in this project:

1. Traditional machine learning models trained on the dataset after having performed dimensionality reduction
2. Deep learning models trained on the original dataset

The traditional machine learning models used include: SVM, Decision Tree, Random Forest, Kth Nearest Neighbor, XGBoost, and AdaBoost. For deep learning, convolutional neural networks (CNNs) were used. The CNNs were trained on the original data. The traditional machine learning models were trained on a dataset that had undergone dimensionality reduction through the use of Principal Component Analysis (PCA). In both approaches, we performed initial training of our models with no parameter tuning. We used the accuracy of these initial models as baseline performance metrics. We then performed hyperparameter optimization through the use of grid search for each of our models with a focus on increasing accuracy. After achieving the highest accuracy for each of our individual models, we compared the performance of all our models using various metrics: accuracy, precision, recall, and AUC to determine the best model.

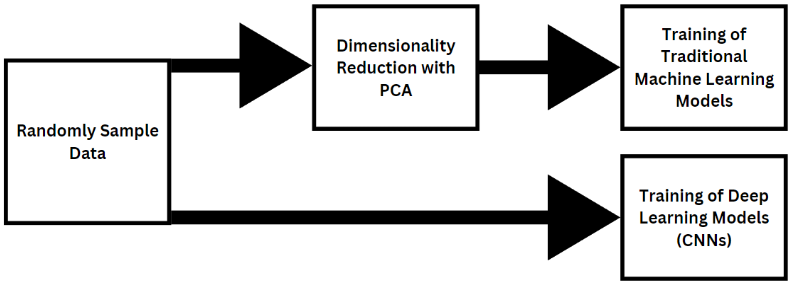

Randomly Sample Data
Dimensionality Reduction with PCA
Training of Traditional Machine Learning Models
Training of Deep Learning Models (CNNs)

Figure 1: Project Workflow

2. Theoretical Background

2.1 Dimensionality Reduction

Dimensionality Reduction is a technique for representing data with fewer features by eliminating redundant and useless parts of the data. Two approaches for dimensionality reduction include feature selection and feature extraction. Feature selection attempts to find the best subset of features whereas feature extraction transforms the data into a lower dimensional space.

Principal Component Analysis (PCA) is a feature extraction technique that uses an orthogonal transformation to convert correlated features into linearly uncorrelated features called principal components.

2.2 Machine Learning Models

Nonlinear SVM

Support Vector Machine is a classification algorithm that attempts to find the best hyperplane in a feature space for the purpose of separating data samples for classification. The best hyperplane is the one with the highest separation, that is the largest distance to the nearest data samples. Nonlinear SVM is a variation of SVM that involves transforming the original data into some higher-dimensional space so that it becomes linearly separable.

Decision Tree

A decision tree is a tree-like classification model wherein each node represents a decision on a specific feature, each branch represents the outcome of that decision, and each leaf node represents the final prediction of the model.

Random Forest

Random forest is a classification technique that works by combining the output of multiple decision trees into a single prediction. With random forest, classification is a 4 step process:

1. A random selection of samples is selected from a dataset.
2. A decision tree is constructed from every sample and every decision tree is used to make a prediction.
3. Voting is performed for every predicted result.
4. The most voted prediction result is picked as the final prediction.

KNN

K-nearest neighbors is a classification algorithm that classifies an object based on the closest samples in the feature space. It operates on the assumption that similar points can be found close to each other. The KNN algorithm is as follows:

1. Initialize K as the chosen number of neighbors.
2. For each example in the data, calculate the Euclidean distance between the query-node and all other example nodes.
3. Sort the distances from smallest to largest distance and pick the first K-entries from this sorted list
4. Identify the labels for each of these K entries
5. Select the mode of these K labels as the label for the query node

Xgboost

XGBoost or eXtreme Gradient Boosting is a predictive model that implements gradient boosted decision trees. It is similar to random forest in that they are both models built on combining multiple decision trees. The difference is that with XGBoost, everytime a decision tree is trained, the weights of the training sample are adjusted based on the training deviation of the model. That is, the weight of correct classification samples are decreased and the weights of misclassified samples are increased. Each subsequent tree learns from the deviation of the previous tree. An accurate model can then be constructed by combining the decision trees.

Adaboost

AdaBoost or Adaptive Boosting is a boosting technique used as an ensemble model. It works by re-assigning weights to each instance, with higher weights assigned to instances that have been incorrectly classified.

CNN

A convolutional neural network is a type of neural network. It typically consists of three types of layers: a convolutional layer for feature extraction, a pooling layer for dimensionality reduction, and a fully connected layer for classification.

2.3 Measurement

Accuracy

The formula for accuracy is given as follows:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{(True\ Positives + True\ Negatives + False\ Positives + False\ negatives)}$$

It is a measurement of how often a model was correct overall.

Precision

The formula for precision is given as follows:

$$Precision\ = \frac{True\ Positives}{(True\ Positives + False\ Positives)}$$

It is a measurement of how good a model is at predicting a specific category.

Recall

The formula for recall is given as follows:

$$Precision\ = \frac{True\ Positives}{(True\ Positives + False\ Negatives)}$$

It is a measure of how many times the model was able to detect a specific category.

ROC Curve and AUC

A Receiver Operating Characteristic Curve (ROC Curve) is a graph that plots true positive rate versus false positive rate. It shows the performance of a classification model at different classification thresholds.

AUC or “area under the ROC Curve” provides a measure of performance across all possible classification thresholds. It represents a measure of separability, that is how well a model is able to distinguish between classes.

3. Methodology

3.1 Data Review

For this project we used the ASL Alphabet dataset hosted on Kaggle. This dataset was separated into two smaller datasets; one for testing and one for training. Because the testing set included only one sample for every class, it was omitted and instead the training dataset was split for use in both testing and training. Included in this training dataset are images separated into 29 folders, one for every class. The 29 classes include the 26 letters of the English alphabet and signs for "del," "nothing," and "space." All 29 folders include 3000 images for a total of 87,000 images. Each image is 200x200 pixels and in color.

Because of computational limitations, we used only images for the 26 letters, omitting images for "del," "nothing," and "space." Moreover, we both randomly sampled 1000 images from each class in the data and decreased the size of the individual images using skimage.transform.resize to 50x50 pixels. This balanced dataset of 26,000 images was used to train our Convolutional Neural Networks directly, but to train our traditional machine learning models, we first performed dimensionality reduction with PCA.

3.2 Dimensionality Reduction with PCA

Despite reducing the original images from 200x200 to 50x50 pixels, in order to train our models in a reasonable amount of time, we performed dimensionality reduction to further reduce the dataset with the use of Sklearn’s built in PCA class. By specifying the desired variance, the class can determine the necessary number of components, that is features. A variance of 95% to 99% is recommended. Because of the large number of required components for a variance of 99%, we used a variance of 95% which only required 80 components. Thus by performing PCA, we were able to reduce the number of features from 50x50x3 = 7500 to 80. This reduced dataset was used for training our traditional machine learning models.

3.3 Hyperparameter Tuning with Gridsearch

The hyperparameter values of a machine learning model can greatly affect its performance. By selecting the correct hyperparameter values, the performance of a model can be improved. Gridsearch provides a systematic method for testing a variety of hyperparameter values in order to determine the best performing values.

Each model has a different set of hyperparameters that can be tuned. For each model, we picked a few and used gridsearch to determine the optimal values. The models and parameters tuned are listed in the following table:

Model	**Hyperparamter(s) Tuned**
CNN	“batch_size”, “dropout_rate”
SVM	“C”, “gamma”
KNN	“n_neighbors”
Random Forest	“max_depth” , “n_estimators”
XGBoost	“max_depth”, “n_estimators”
Decision Tree	“criterion”, “min_samples_leaf”, “min_samples_split” “max_depth”
AdaBoost	“learning_rate”, “n_estimators”

Figure 2: Model Hyperparameters

Because of computational constraints, only a few hyperparameters were picked for each model. For each model, an initial gridsearch with widely spaced values was used, then a more granular search.

4. Results

The initial training of our models was performed with no parameter tuning. Hyperparameter optimization through the use of grid search was then performed to increase accuracy. Moreover, in the case of our convolutional neural networks, deeper, more complex models were tested.

The first algorithm used was a nonlinear support vector machine (SVM) with a Gaussian Radial-Basis Function (RBF) kernel. Using grid search, it was determined the best parameter values for “C” and “gamma” were 10 and .001, respectively. The accuracy of SVC Grid was increased from 84.55% to 93.23%.

The second algorithm we used is Decision Tree, along with grid search to find the best parameter values. After performing a grid search for the Decision Tree, it was determined the best parameter values for “criterion”, “min_samples_leaf”, and “min_samples_split” were “entropy”, 1, and 2, respectively. The accuracy for this model is 62.52%, which is only a slight increase from the initial decision tree accuracy of 62.24%.

The next algorithm is Random Forest with 19 trees. The initial accuracy of Random Forest was 82.21%. After performing grid search, the accuracy increased to 88.49% with the best parameter values for “max_depth” and “n_estimators” being 40 and 300, respectively.

For the K Nearest Neighbor (KNN) algorithm, the model was initially trained with k=3, and achieved an accuracy of 84.8%. After grid search, the best accuracy achieved was 85.17%, with a parameter value for “n_neighbors” of 1. The best accuracy achieved by KNN after performing gridsearch was 91.38%.

Afterwards, we first used XGBoost with 29 gradient boosted trees. The initial accuracy for XGBoost was 82.09%. Using grid search the best parameter values for “max_depth” and “n_estimators” were found to be, respectively, 25 and 70. The accuracy of XGBoost with these values is 84.89%.

For AdaBoost, 29 estimators were used initially, achieving an accuracy of 13.2%. After grid search, the best parameter values for “learning_rate” and “n_estimators” were found to be 1.0 and 200. The accuracy for AdaBoost with these values was increased to 17.72%.

When using Convolutional Neural Network (CNN), a shallow network was used initially, with 32 filters in the convolutional 2D layer, a kernel size of 5 by 5, max pooling size of 2 by 2, and rectified linear unit (RELU) for the activation function. Then, a flatten layer, a dense layer with the size of twice the image size, another dense layer with the size of 26 for the number of labels. The structure gave an accuracy of 91.15%.

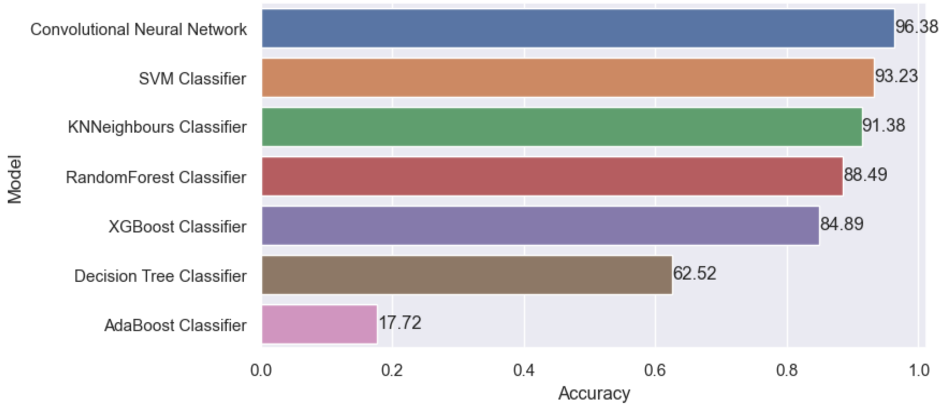
Through experimentation, a deeper CNN was constructed. This network included 2 convolutional 2D layers, each with 32 5x5 filters. This was followed by a max pooling layer with size 2x2, then a dropout layer to drop 25% of the weights, a flatten layer, a dense layer with 128 units, another dropout 25% layer, and the final dense layer with 26 units. This deeper network achieved an accuracy of 96.38%. The highest among all models.

The results of these initial and optimized models are listed in the following table.

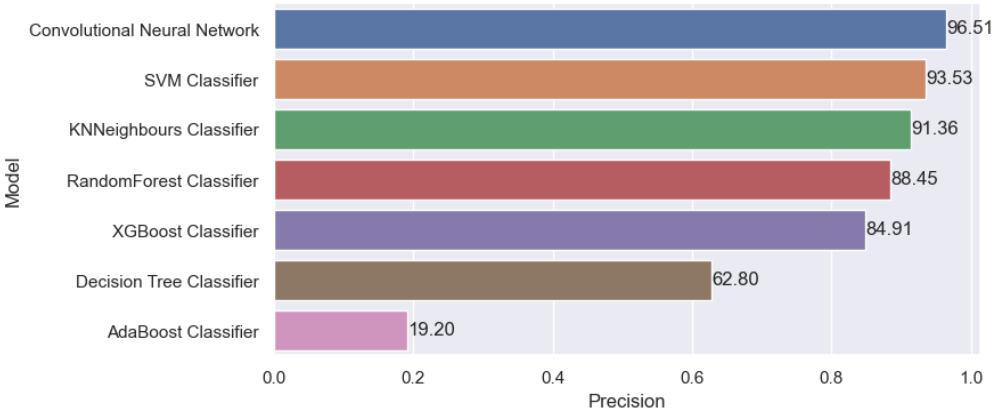
Model Used	Initial Accuracy	Final Accuracy
CNN	91.15	96.38
SVM	84.55	93.23
KNN	85.17	91.36
Random Forest	87.21	88.49
XGBoost	82.09	84.89
Decision Tree	62.24	62.52
AdaBoost	13.20	17.72

Figure 3: Model Accuracy

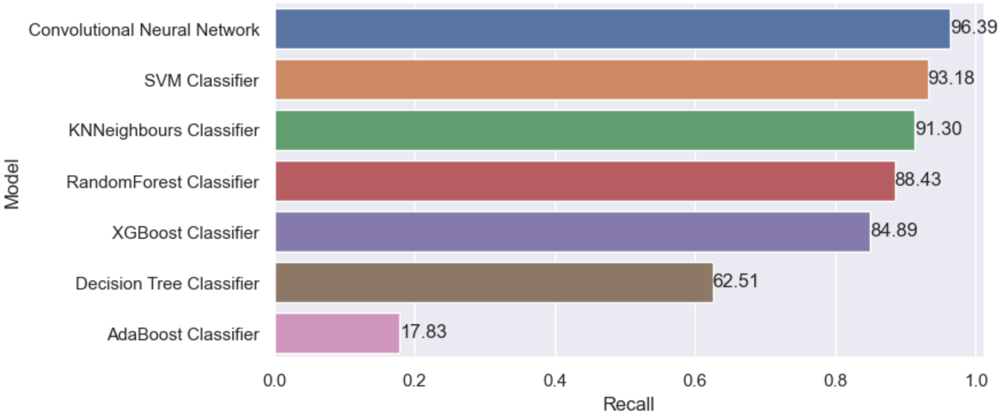
All models saw an improvement in accuracy, though some more than others. CNN had both the highest initial and final accuracy of all the models with 91.15% and 96.38% respectively. SVM is the model that benefited the most from hyperparameter tuning with an increase of almost 9% in accuracy. This final accuracy of 93.23% was the highest among the traditional machine learning models. Decision Tree saw the smallest change in accuracy with an increase of only 0.28%, from 62.24% to 62.52%. Although not listed, many of the values tested resulted in lower accuracy than the initial model for decision tree. Adaboost had the lowest accuracy amongst all the models both in the initial and final accuracy. Despite trying a variety of parameter values, AdaBoost could only achieve a final accuracy of 17.17%, significantly lower than the next lowest value of decision tree's 62.52%. The final, highest accuracies are listed in descending order in the following bar chart.



After the highest accuracy was achieved for the individual models, other metrics were considered. The first of which being precision. We list the precisions for the highest accuracy models in the following bar chart.

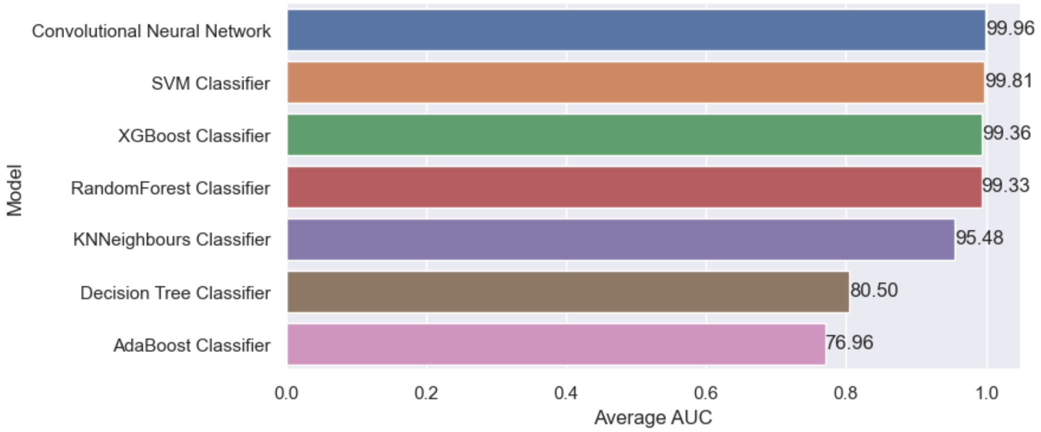


The precision values for our models in the above chart are listed from highest to lowest. We note that the ordering is identical to that of the accuracy chart, that is, the most accurate models were also the most precise. The convolutional neural network scored the highest overall with 96.51%. SVM scored the second highest overall, and the highest amongst the traditional neural networks with 93.53%. The rest of the models scored similarly between the mid 80's and lower 90's except for decision tree which scored lower at 62.80% and AdaBoost which scored the lowest still with 19.20%. The next metric we considered was recall, which we list in the following chart.



Again we see the ordering aligns with model accuracy. The convolutional neural network had the highest overall recall score of 96.39%. SVM had the second highest overall and highest among the traditional machine learning models with a score of 93.18%. The rest of the models scored similarly except for Decision Tree and AdaBoost which scored significantly lower with 62.51% and 17.83% respectively.

The final metric we considered was AUC, the area under the ROC Curve. Because our project required multi-class classification, we took a "one-versus-rest" approach for each class when developing the ROC Curve, and then calculated the average AUC between all the classes. We list these average AUC values for each model in the following chart.



Again the highest scoring overall was CNN with 99.96 and the second highest overall and highest amongst the traditional machine learning models was SVM with 99.81. The majority of models achieved a high average AUC score with the top five scoring above 95 and the top four, over 99. As an example we include the ROC Curve for the Random Forest model.

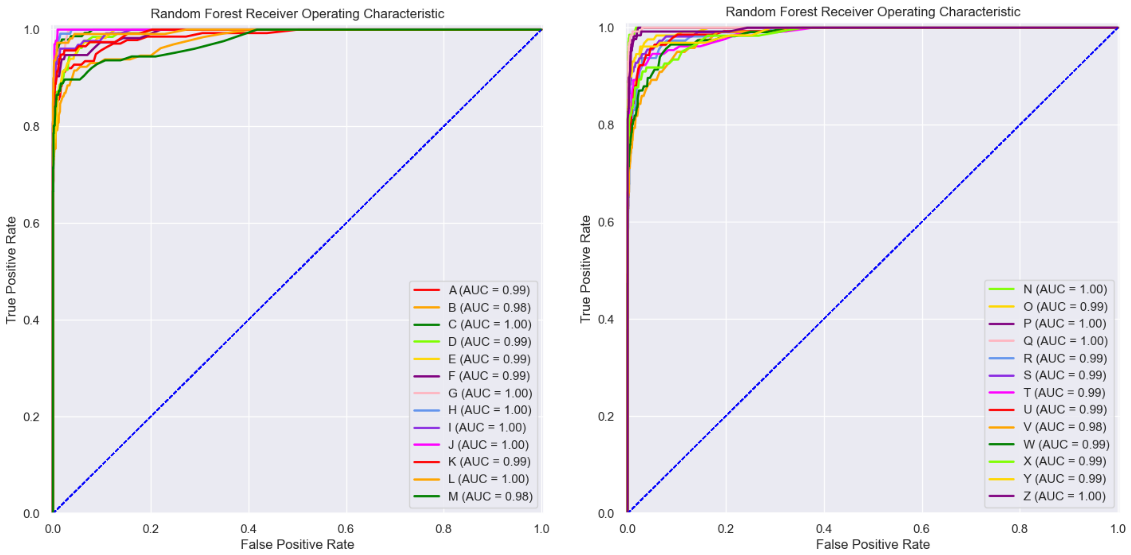
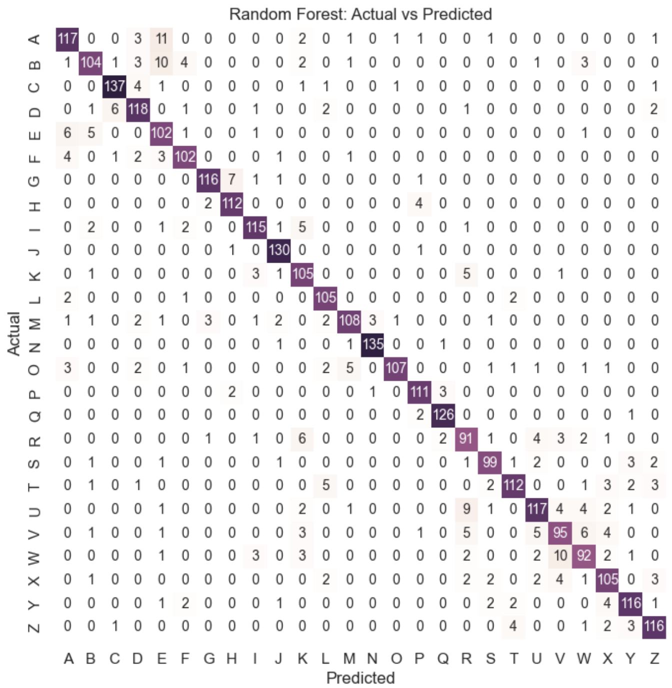


Figure 4: ROC Curve

Although Random Forest only achieved an accuracy of 88.49%, because there are 26 classes, this translates to a high level of accuracy for each individual class and thus a high value for average AUC. We see this in the confusion matrix for Random Forest.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	117	0	0	3	11	0	0	0	0	0	2	0	1	0	1	1	0	0	1	0	0	0	0	0	0	1
B	1	104	1	3	10	4	0	0	0	0	2	0	1	0	0	0	0	0	0	0	1	0	3	0	0	0
C	0	0	137	4	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1
D	0	1	6	118	0	1	0	0	1	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	2
E	6	5	0	0	102	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
F	4	0	1	2	3	102	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	116	7	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	2	112	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
I	0	2	0	0	1	2	0	0	115	1	5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	1	0	130	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
K	0	1	0	0	0	0	0	0	3	1	105	0	0	0	0	0	0	5	0	0	0	1	0	0	0	0
L	2	0	0	0	0	1	0	0	0	0	0	105	0	0	0	0	0	0	0	2	0	0	0	0	0	0
M	1	1	0	2	1	0	3	0	1	2	0	2	108	3	1	0	0	0	1	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	1	0	0	1	135	0	0	1	0	0	0	0	0	0	0	0	0
O	3	0	0	2	0	1	0	0	0	0	0	2	5	0	107	0	0	0	1	1	1	0	1	1	0	0
P	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	111	3	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	126	0	0	0	0	0	0	0	1	0
R	0	0	0	0	0	0	1	0	1	0	6	0	0	0	0	0	2	91	1	0	4	3	2	1	0	0
S	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	99	1	2	0	0	0	3	2
T	0	1	0	1	0	0	0	0	0	0	0	5	0	0	0	0	0	0	2	112	0	0	1	3	2	3
U	0	0	0	0	1	0	0	0	0	0	2	0	1	0	0	0	0	9	1	0	117	4	4	2	1	0
V	0	1	0	0	1	0	0	0	0	0	3	0	0	0	0	1	0	5	0	0	5	95	6	4	0	0
W	0	0	0	0	1	0	0	0	3	0	3	0	0	0	0	0	0	2	0	0	2	10	92	2	1	0
X	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	2	0	2	4	1	105	0	3
Y	0	0	0	0	1	2	0	0	0	1	0	0	0	0	0	0	0	0	2	2	0	0	0	4	116	1
Z	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	1	2	3	116

Figure 5: Random Forest Confusion Matrix

We see high values listed along the diagonal meaning that the model could accurately identify what letter each individual image was displaying. The entries along the diagonal with the lowest values are “R”, “S”,”V”, and “W.” This means the model had the hardest time identifying those four letters. When comparing the performance of the other models, we see similar results; those four classes were the most difficult to identify. We include ROC Curves and confusion matrices for the other models at the end of the report.

We summarize the above results by listing all metrics for every model together in the following chart.

Model	**Accuracy**	**Precision**	**Recall**	**Average AUC**
CNN	96.38	96.65	96.38	99.96
SVM	93.23	93.53	93.17	99.81
KNN	91.38	91.36	91.29	95.54
Random Forest	88.49	88.45	88.43	99.33
XGBoost	84.89	84.90	84.89	99.36
Decision Tree	62.52	62.79	62.50	80.50
AdaBoost	17.72	19.20	17.82	76.96

Figure 6: Comparison of Model Performance

We see that for every metric, that is accuracy, precision, recall, and the average AUC, CNN scored the highest overall. For every metric, SVM scored the second best overall and best among the traditional machine learning models. KNN, Random Forest, and XGBoost scored reasonably well on every metric, and Decision Tree and AdaBoost scored noticeably worse than the other models on every metric.

5. Conclusion and Future Work

5.1 Conclusion

Our goal for this project was developing a well performing model for the purpose of identifying images of the different letters of the alphabet in American Sign Language. In order to do so, we trained a variety of machine learning models, both traditional machine learning and deep learning models. We then performed hyperparameter tuning to achieve the highest accuracy for each model. Additionally, for our CNN, we tested a variety of model structures. After doing so, we considered a variety of metrics to determine the best performing model. By doing so, we determined that our CNN was the best model for image classification of the ASL Alphabet. For every metric, accuracy, precision, recall, and the average AUC, CNN scored the highest overall. Between the 26 classes, that is the 26 letters of the ASL Alphabet, it could correctly identify an image with an accuracy of 96.38%.

5.2 Suggestions for Future Work

Due mostly to computational limitations, there are several ways this work can be expanded and improved.

1. Training models on the entire data

The original dataset consisted of 3000 images for every class. We randomly sampled 1000 from every class for use in training. Training on a larger dataset, particularly in deep learning can result in increased performance.

2. Use more components in PCA for retaining higher variance

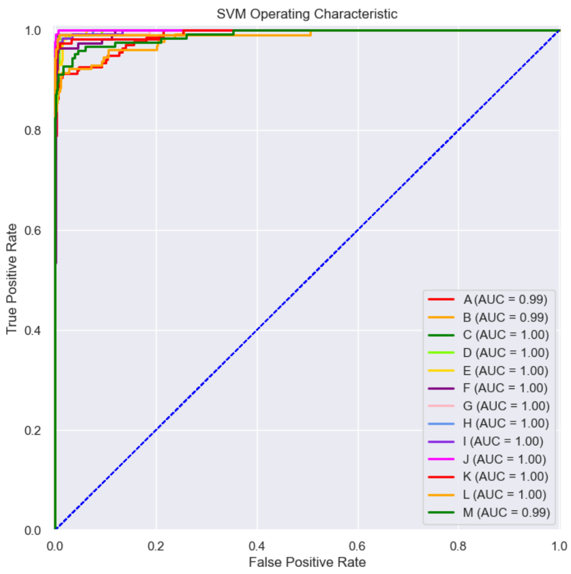
When performing PCA, it is recommended that one chooses k number of features such that 95% to 99% of the variance is preserved. In order to make the training more efficient, we picked a variance of 95%. By picking 99%, more of the information can be retained and model performance improved.

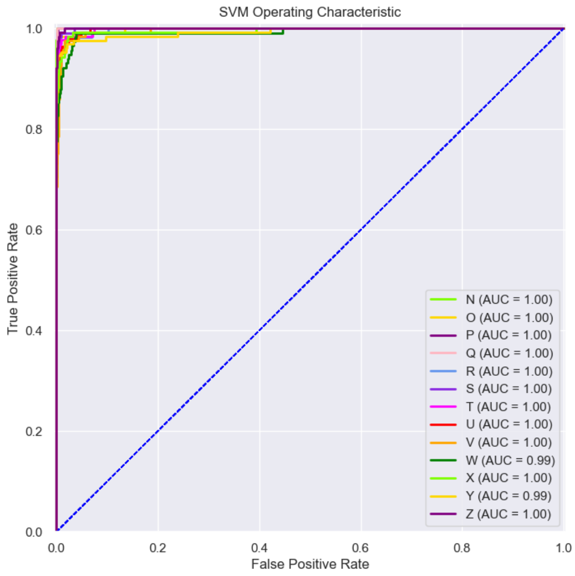
3. Perform a more expansive gridsearch

Because gridsearch works by training multiple models with different parameter values, it is very computationally expensive and time consuming. Because of this, we were limited in the number of parameters we could include, and the number of values for each parameter. By performing gridsearch with a larger number of parameters and values, we could find more optimal, better performing models.

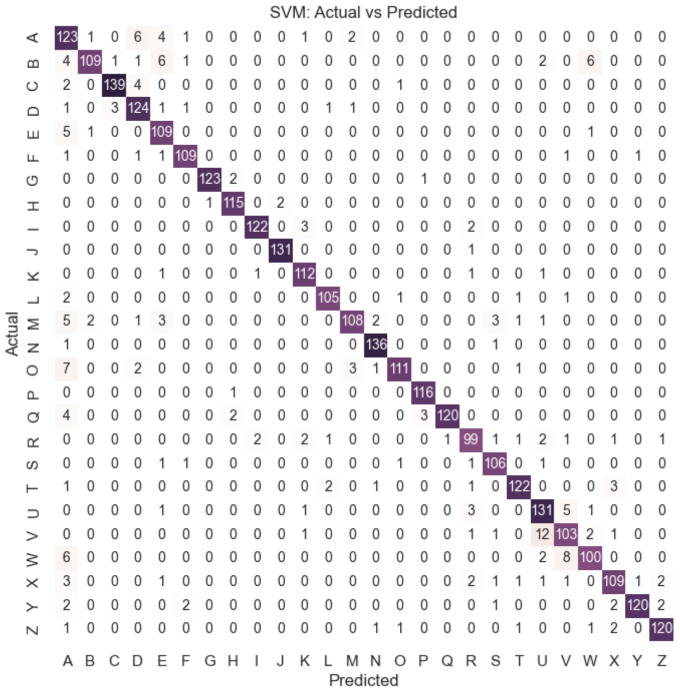
6. Model ROC Curves and Confusion Matrices

SVM Roc Curve:



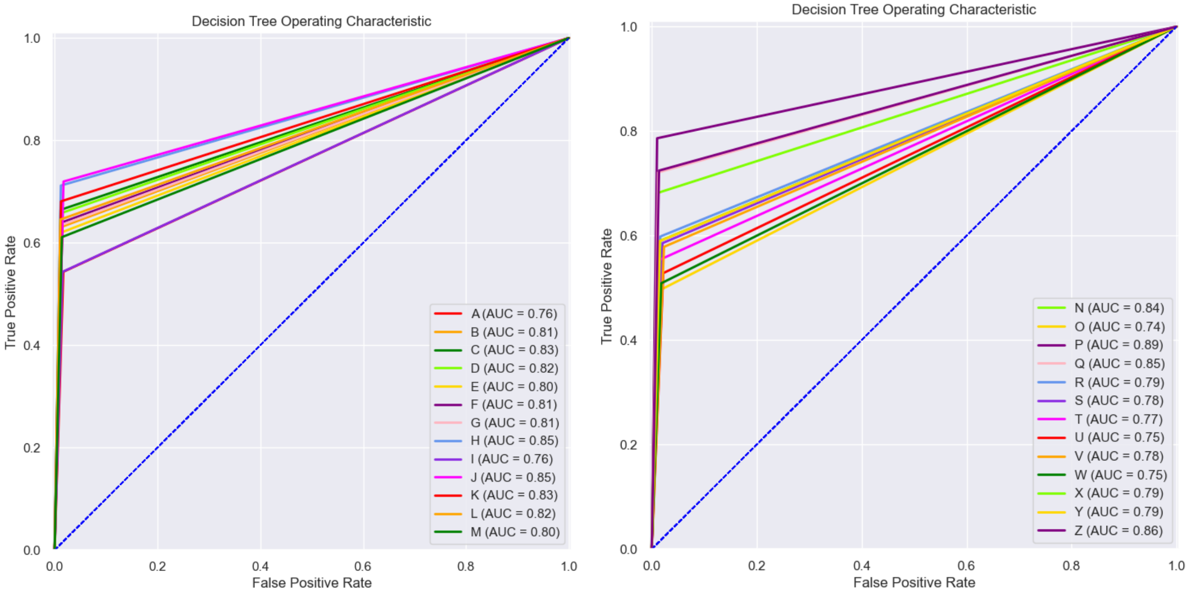


SVM Confusion Matrix:

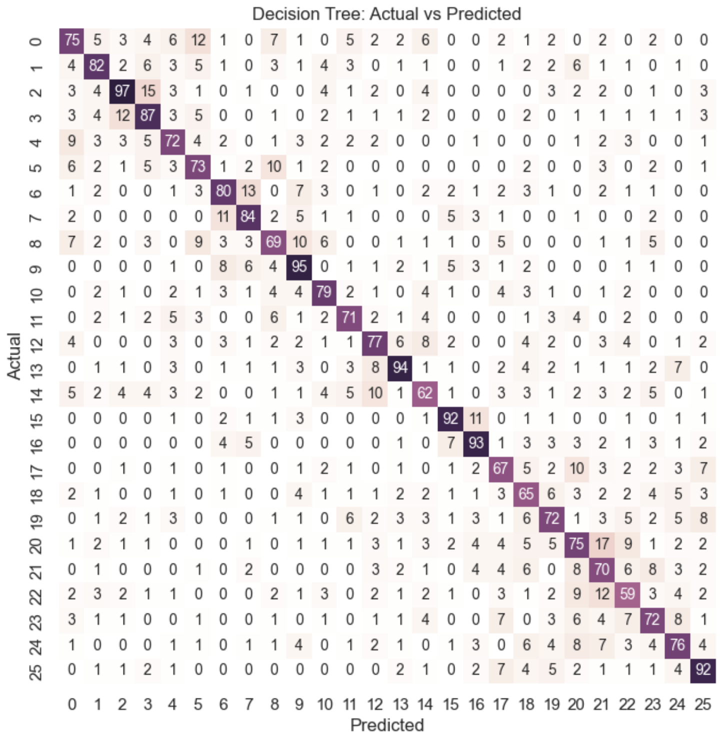


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	123	1	0	6	4	1	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
B	4	109	1	1	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	6	0	0	0
C	2	0	139	4	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D	1	0	3	124	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
E	5	1	0	0	109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
F	1	0	0	1	1	109	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
G	0	0	0	0	0	0	123	2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1	115	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	122	0	3	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	131	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
K	0	0	0	0	1	0	0	0	1	0	112	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
L	2	0	0	0	0	0	0	0	0	0	0	105	0	0	1	0	0	0	0	1	0	1	0	0	0	0
M	5	2	0	1	3	0	0	0	0	0	0	0	108	2	0	0	0	0	3	1	1	0	0	0	0	0
N	1	0	0	0	0	0	0	0	0	0	0	0	0	136	0	0	0	0	1	0	0	0	0	0	0	0
O	7	0	0	2	0	0	0	0	0	0	0	0	3	1	111	0	0	0	0	1	0	0	0	0	0	0
P	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	116	0	0	0	0	0	0	0	0	0	0
Q	4	0	0	0	0	0	0	2	0	0	0	0	0	0	0	3	120	0	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	2	0	2	1	0	0	0	0	1	99	1	1	2	1	0	1	0	1
S	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	106	0	1	0	0	0	0	0
T	1	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	1	0	122	0	0	0	3	0	0
U	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0	131	5	1	0	0	0
V	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	12	103	2	1	0	0
W	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	8	100	0	0	0
X	3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	1	1	1	1	0	109	1	2
Y	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	120	2
Z	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	1	2	0	120

Decision Tree Roc Curve:



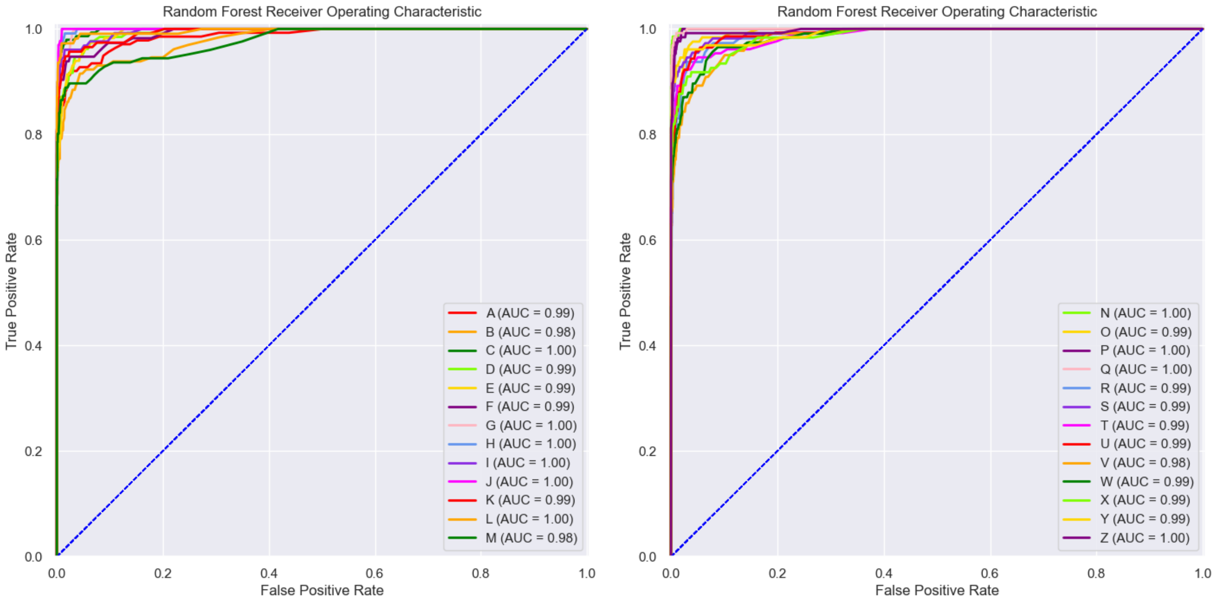
Decision Tree Confusion Matrix:



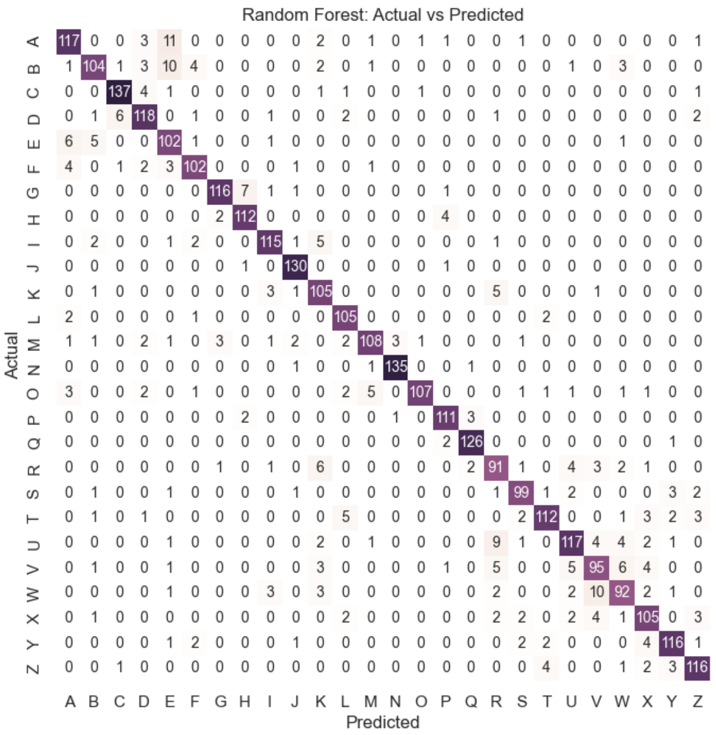
Decision Tree: Actual vs Predicted

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	75	5	3	4	6	12	1	0	7	1	0	5	2	2	6	0	0	2	1	2	0	2	0	2	0	0
1	4	82	2	6	3	5	1	0	3	1	4	3	0	1	1	0	0	1	2	2	6	1	1	0	1	0
2	3	4	97	15	3	1	0	1	0	0	4	1	2	0	4	0	0	0	0	3	2	2	0	1	0	3
3	3	4	12	87	3	5	0	0	1	0	2	1	1	1	2	0	0	0	2	0	1	1	1	1	1	3
4	9	3	3	5	72	4	2	0	1	3	2	2	2	0	0	0	1	0	0	0	1	2	3	0	0	1
5	6	2	1	5	3	73	1	2	10	1	2	0	0	0	0	0	0	0	2	0	0	3	0	2	0	1
6	1	2	0	0	1	3	80	13	0	7	3	0	1	0	2	2	1	2	3	1	0	2	1	1	0	0
7	2	0	0	0	0	0	11	84	2	5	1	1	0	0	0	5	3	1	0	0	1	0	0	2	0	0
8	7	2	0	3	0	9	3	3	69	10	6	0	0	1	1	1	0	5	0	0	0	1	1	5	0	0
9	0	0	0	0	1	0	8	6	4	95	0	1	1	2	1	5	3	1	2	0	0	0	1	1	0	0
10	0	2	1	0	2	1	3	1	4	4	79	2	1	0	4	1	0	4	3	1	0	1	2	0	0	0
11	0	2	1	2	5	3	0	0	6	1	2	71	2	1	4	0	0	0	1	3	4	0	2	0	0	0
12	4	0	0	0	3	0	3	1	2	2	1	1	77	6	8	2	0	0	4	2	0	3	4	0	1	2
13	0	1	1	0	3	0	1	1	1	3	0	3	8	94	1	1	0	2	4	2	1	1	1	2	7	0
14	5	2	4	4	3	2	0	0	1	1	4	5	10	1	62	1	0	3	3	1	2	3	2	5	0	1
15	0	0	0	0	1	0	2	1	1	3	0	0	0	0	1	92	11	0	1	1	0	0	1	0	1	1
16	0	0	0	0	0	0	4	5	0	0	0	0	0	1	0	7	93	1	3	3	3	2	1	3	1	2
17	0	0	1	0	1	0	1	0	0	1	2	1	0	1	0	1	2	67	5	2	10	3	2	2	3	7
18	2	1	0	0	1	0	1	0	0	4	1	1	1	2	2	1	1	3	65	6	3	2	2	4	5	3
19	0	1	2	1	3	0	0	0	1	1	0	6	2	3	3	1	3	1	6	72	1	3	5	2	5	8
20	1	2	1	1	0	0	0	1	0	1	1	1	3	1	3	2	4	4	5	5	75	17	9	1	2	2
21	0	1	0	0	0	1	0	2	0	0	0	0	3	2	1	0	4	4	6	0	8	70	6	8	3	2
22	2	3	2	1	1	0	0	0	2	1	3	0	2	1	2	1	0	3	1	2	9	12	59	3	4	2
23	3	1	1	0	0	1	0	0	1	0	1	0	1	1	4	0	0	7	0	3	6	4	7	72	8	1
24	1	0	0	0	1	1	0	1	1	4	0	1	2	1	0	1	3	0	6	4	8	7	3	4	76	4
25	0	1	1	2	1	0	0	0	0	0	0	0	0	2	1	0	2	7	4	5	2	1	1	1	4	92

Random Forest Roc Curve:

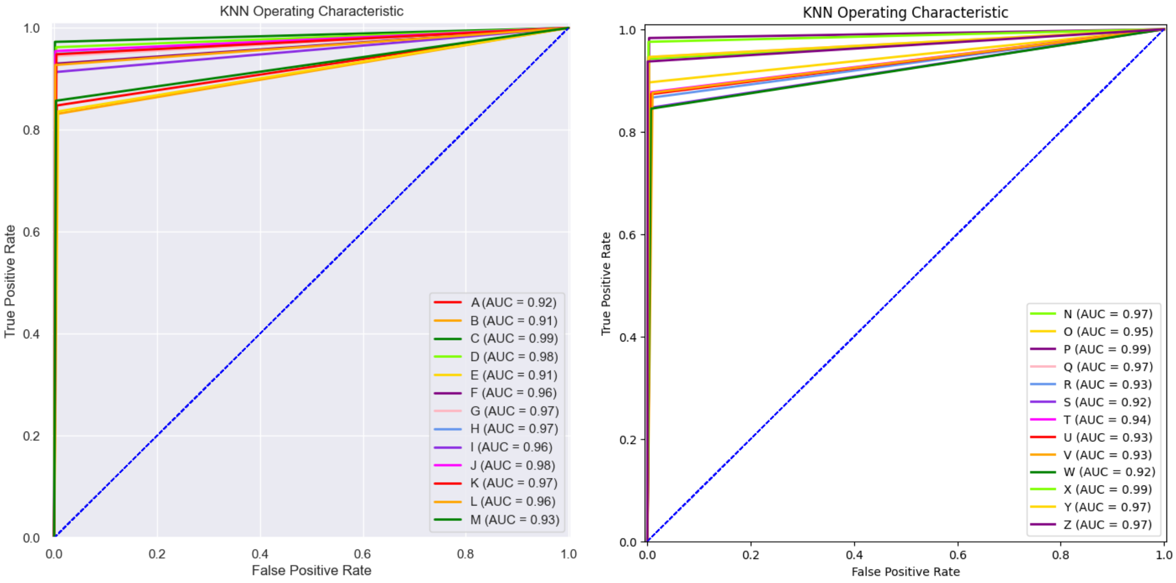


Random Forest Confusion Matrix:

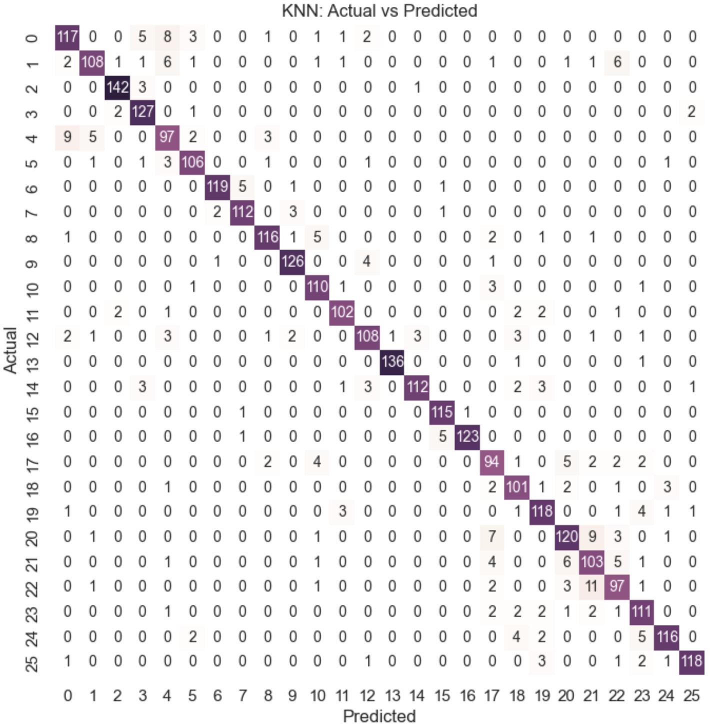


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	117	0	0	3	11	0	0	0	0	0	2	0	1	0	1	1	0	0	1	0	0	0	0	0	0	1
B	1	104	1	3	10	4	0	0	0	0	2	0	1	0	0	0	0	0	0	0	1	0	3	0	0	0
C	0	0	137	4	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1
D	0	1	6	118	0	1	0	0	1	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	2
E	6	5	0	0	102	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
F	4	0	1	2	3	102	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	116	7	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	2	112	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
I	0	2	0	0	1	2	0	0	115	1	5	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	1	0	130	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
K	0	1	0	0	0	0	0	0	3	1	105	0	0	0	0	0	0	5	0	0	0	1	0	0	0	0
L	2	0	0	0	0	1	0	0	0	0	0	105	0	0	0	0	0	0	0	2	0	0	0	0	0	0
M	1	1	0	2	1	0	3	0	1	2	0	2	108	3	1	0	0	0	1	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	1	0	0	1	135	0	0	1	0	0	0	0	0	0	0	0	0
O	3	0	0	2	0	1	0	0	0	0	0	2	5	0	107	0	0	0	1	1	1	0	1	1	0	0
P	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	111	3	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	126	0	0	0	0	0	0	0	1	0
R	0	0	0	0	0	0	1	0	1	0	6	0	0	0	0	0	2	91	1	0	4	3	2	1	0	0
S	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	99	1	2	0	0	0	3	2
T	0	1	0	1	0	0	0	0	0	0	0	5	0	0	0	0	0	0	2	112	0	0	1	3	2	3
U	0	0	0	0	1	0	0	0	0	0	2	0	1	0	0	0	0	9	1	0	117	4	4	2	1	0
V	0	1	0	0	1	0	0	0	0	0	3	0	0	0	0	1	0	5	0	0	5	95	6	4	0	0
W	0	0	0	0	1	0	0	0	3	0	3	0	0	0	0	0	0	2	0	0	2	10	92	2	1	0
X	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	2	0	2	4	1	105	0	3
Y	0	0	0	0	1	2	0	0	0	1	0	0	0	0	0	0	0	0	2	2	0	0	0	4	116	1
Z	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	1	2	3	116

KNN Roc Curve:

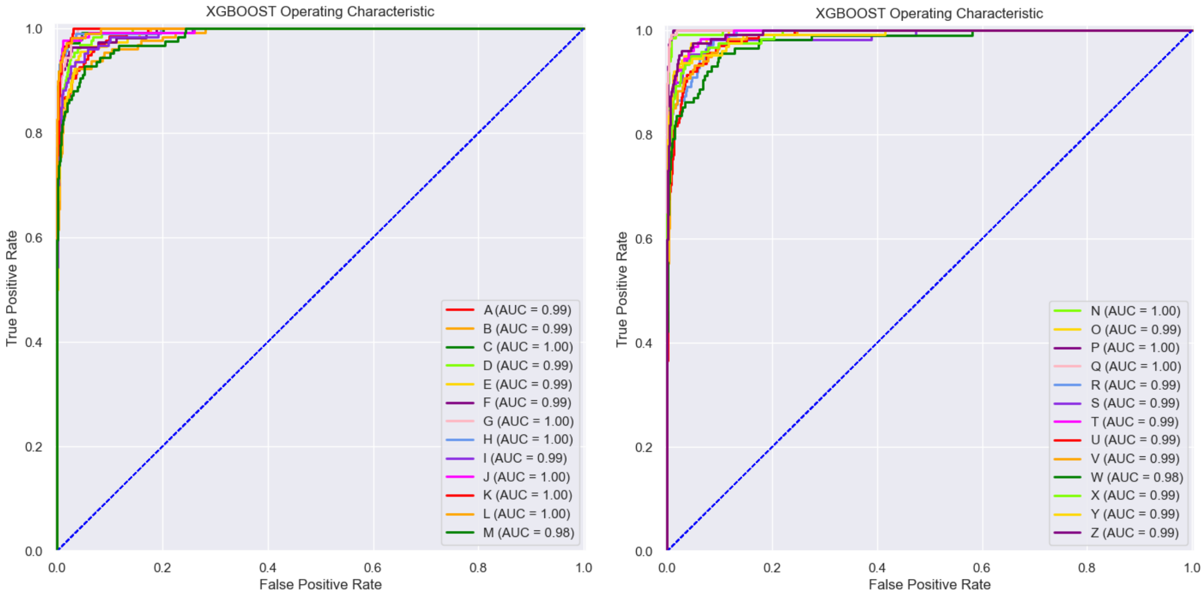


KNN Confusion Matrix:

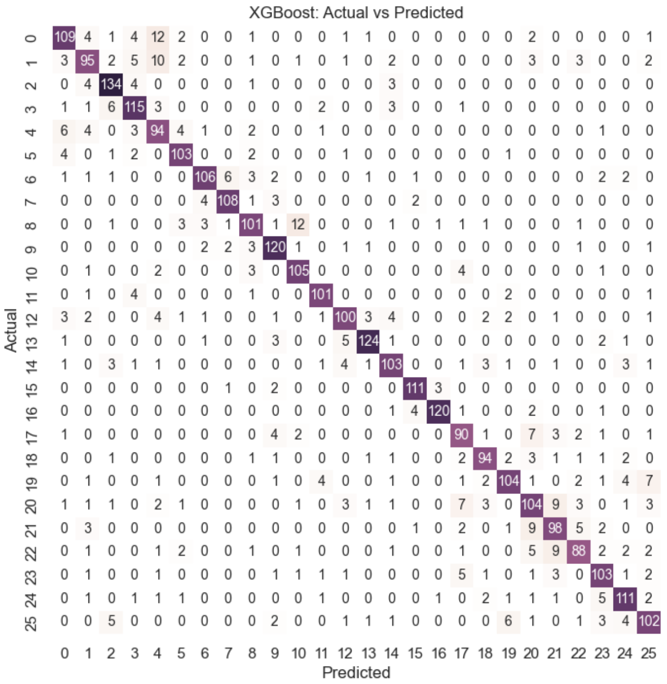


	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	117	0	0	5	8	3	0	0	1	0	1	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0
1	2	108	1	1	6	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1	6	0	0	0
2	0	0	142	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
3	0	0	2	127	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
4	9	5	0	0	97	2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	3	106	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
6	0	0	0	0	0	0	119	5	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	2	112	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
8	1	0	0	0	0	0	0	0	116	1	5	0	0	0	0	0	0	2	0	1	0	1	0	0	0	0
9	0	0	0	0	0	0	1	0	0	126	0	0	4	0	0	0	0	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	110	1	0	0	0	0	0	3	0	0	0	0	0	1	0	0
11	0	0	2	0	1	0	0	0	0	0	0	102	0	0	0	0	0	0	2	2	0	0	1	0	0	0
12	2	1	0	0	3	0	0	0	1	2	0	0	108	1	3	0	0	0	3	0	0	1	0	1	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	136	0	0	0	0	1	0	0	0	0	1	0	0
14	0	0	0	3	0	0	0	0	0	0	0	1	3	0	112	0	0	0	2	3	0	0	0	0	0	1
15	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	115	1	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	123	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	2	0	4	0	0	0	0	0	0	94	1	0	5	2	2	2	0	0
18	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	101	1	2	0	1	0	3	0
19	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	118	0	0	1	4	1	1
20	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	7	0	0	120	9	3	0	1	0
21	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	6	103	5	1	0	0
22	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	3	11	97	1	0	0
23	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	1	2	1	111	0	0
24	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	4	2	0	0	0	5	116	0
25	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	3	0	0	1	2	1	118

XGBoost Roc Curve:

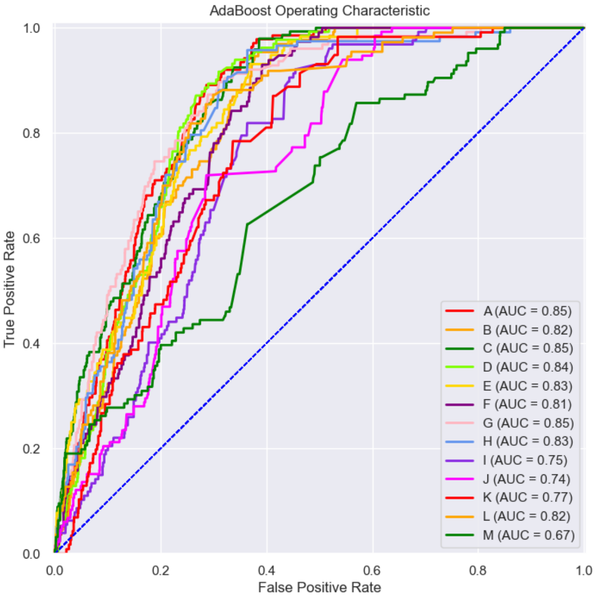


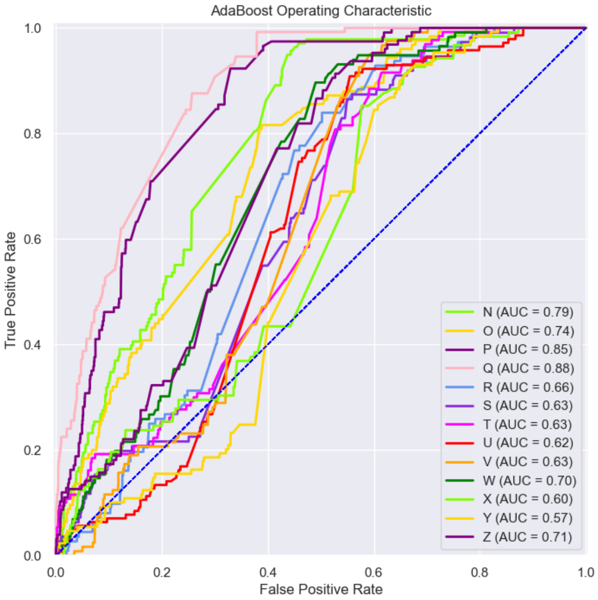
XGBoost Confusion Matrix:


XGBoost: Actual vs Predicted

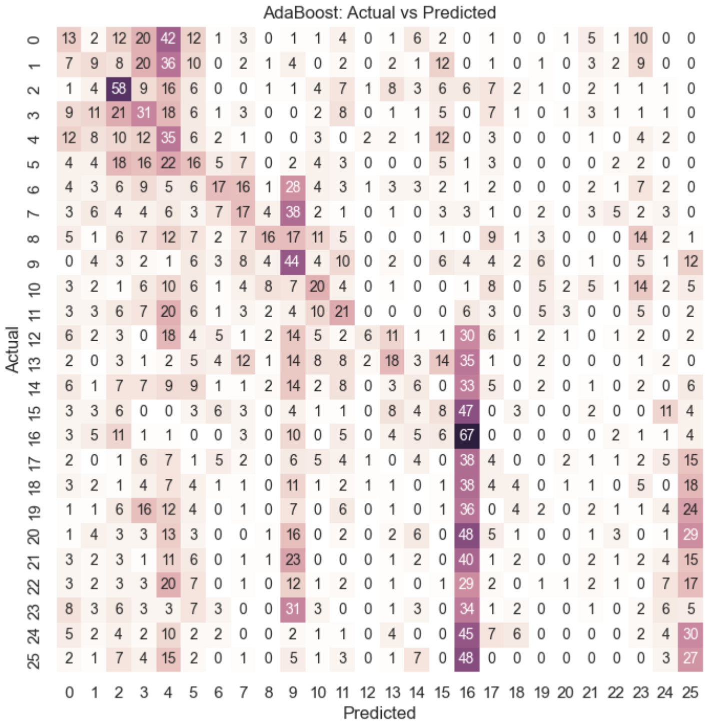
Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	109	4	1	4	12	2	0	0	1	0	0	0	1	1	0	0	0	0	0	0	2	0	0	0	0	1
1	3	95	2	5	10	2	0	0	1	0	1	0	1	0	2	0	0	0	0	0	3	0	3	0	0	2
2	0	4	134	4	0	0	0	0	1	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
3	1	1	6	115	3	0	0	0	0	0	0	2	0	0	3	0	0	1	0	0	0	0	0	0	0	0
4	6	4	0	3	94	4	1	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
5	4	0	1	2	0	103	0	0	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
6	1	1	1	0	0	0	106	6	3	2	0	0	0	1	0	1	0	0	0	0	0	0	0	2	2	0
7	0	0	0	0	0	0	4	108	1	3	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
8	0	0	1	0	0	3	3	1	101	1	12	0	0	0	1	0	1	1	1	0	0	0	1	0	0	0
9	0	0	0	0	0	0	2	2	3	120	1	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1
10	0	1	0	0	2	0	0	0	3	0	105	0	0	0	0	0	0	4	0	0	0	0	0	1	0	0
11	0	1	0	4	0	0	0	0	1	0	0	101	0	0	0	0	0	0	0	2	0	0	0	0	0	1
12	3	2	0	0	4	1	1	0	0	1	0	1	100	3	4	0	0	0	2	2	0	1	0	0	0	1
13	1	0	0	0	0	0	1	0	0	3	0	0	5	124	1	0	0	0	0	0	0	0	0	2	1	0
14	1	0	3	1	1	0	0	0	0	0	0	1	4	1	103	0	0	1	3	1	0	1	0	0	3	1
15	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0	111	3	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	120	1	0	0	2	0	0	1	0	0
17	1	0	0	0	0	0	0	0	0	4	2	0	0	0	0	0	0	90	1	0	7	3	2	1	0	1
18	0	0	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	2	94	2	3	1	1	1	2	0
19	0	1	0	0	1	0	0	0	0	1	0	4	0	0	1	0	0	1	2	104	1	0	2	1	4	7
20	1	1	1	0	2	1	0	0	0	0	1	0	3	1	1	0	0	7	3	0	104	9	3	0	1	3
21	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	1	9	98	5	2	0	0
22	0	1	0	0	1	2	0	0	1	0	1	0	0	0	1	0	0	1	0	0	5	9	88	2	2	2
23	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	5	1	0	1	3	0	103	1	2
24	0	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	1	0	2	1	1	1	0	5	111	2
25	0	0	5	0	0	0	0	0	0	2	0	0	1	1	1	0	0	0	0	6	1	0	1	3	4	102

AdaBoost Roc Curve:



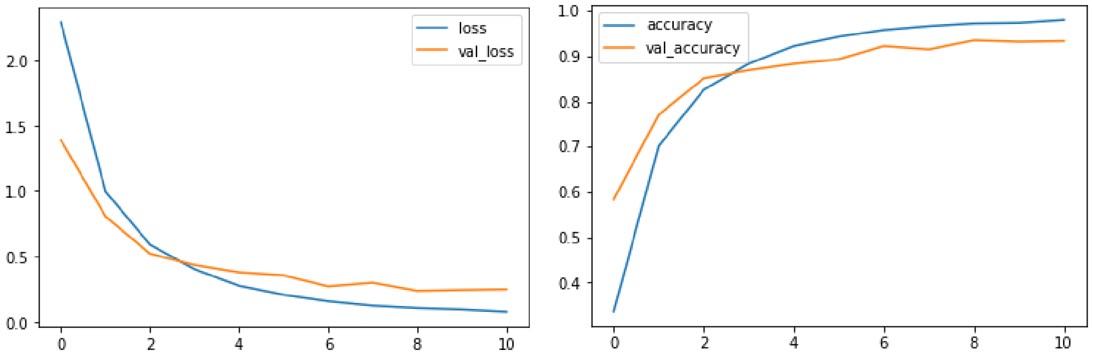


AdaBoost Confusion Matrix:

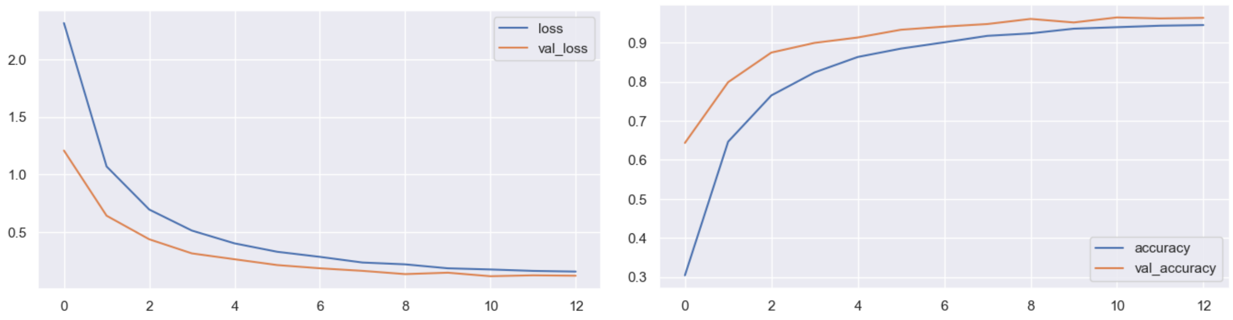


Actual \ Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	13	2	12	20	42	12	1	3	0	1	1	4	0	1	6	2	0	1	0	0	1	5	1	10	0	0
1	7	9	8	20	36	10	0	2	1	4	0	2	0	2	1	12	0	1	0	1	0	3	2	9	0	0
2	1	4	58	9	16	6	0	0	1	1	4	7	1	8	3	6	6	7	2	1	0	2	1	1	1	0
3	9	11	21	31	18	6	1	3	0	0	2	8	0	1	1	5	0	7	1	0	1	3	1	1	1	0
4	12	8	10	12	35	6	2	1	0	0	3	0	2	2	1	12	0	3	0	0	0	1	0	4	2	0
5	4	4	18	16	22	16	5	7	0	2	4	3	0	0	0	5	1	3	0	0	0	0	2	2	0	0
6	4	3	6	9	5	6	17	16	1	28	4	3	1	3	3	2	1	2	0	0	0	2	1	7	2	0
7	3	6	4	4	6	3	7	17	4	38	2	1	0	1	0	3	3	1	0	2	0	3	5	2	3	0
8	5	1	6	7	12	7	2	7	16	17	11	5	0	0	0	1	0	9	1	3	0	0	0	14	2	1
9	0	4	3	2	1	6	3	8	4	44	4	10	0	2	0	6	4	4	2	6	0	1	0	5	1	12
10	3	2	1	6	10	6	1	4	8	7	20	4	0	1	0	0	1	8	0	5	2	5	1	14	2	5
11	3	3	6	7	20	6	1	3	2	4	10	21	0	0	0	0	6	3	0	5	3	0	0	5	0	2
12	6	2	3	0	18	4	5	1	2	14	5	2	6	11	1	1	30	6	1	2	1	0	1	2	0	2
13	2	0	3	1	2	5	4	12	1	14	8	8	2	18	3	14	35	1	0	2	0	0	0	1	2	0
14	6	1	7	7	9	9	1	1	2	14	2	8	0	3	6	0	33	5	0	2	0	1	0	2	0	6
15	3	3	6	0	0	3	6	3	0	4	1	1	0	8	4	8	47	0	3	0	0	2	0	0	11	4
16	3	5	11	1	1	0	0	3	0	10	0	5	0	4	5	6	67	0	0	0	0	0	2	1	1	4
17	2	0	1	6	7	1	5	2	0	6	5	4	1	0	4	0	38	4	0	0	2	1	1	2	5	15
18	3	2	1	4	7	4	1	1	0	11	1	2	1	1	0	1	38	4	4	0	1	1	0	5	0	18
19	1	1	6	16	12	4	0	1	0	7	0	6	0	1	0	1	36	0	4	2	0	2	1	1	4	24
20	1	4	3	3	13	3	0	0	1	16	0	2	0	2	6	0	48	5	1	0	0	1	3	0	1	29
21	3	2	3	1	11	6	0	1	1	23	0	0	0	1	2	0	40	1	2	0	0	2	1	2	4	15
22	3	2	3	3	20	7	0	1	0	12	1	2	0	1	0	1	29	2	0	1	1	2	1	0	7	17
23	8	3	6	3	3	7	3	0	0	31	3	0	0	1	3	0	34	1	2	0	0	1	0	2	6	5
24	5	2	4	2	10	2	2	0	0	2	1	1	0	4	0	0	45	7	6	0	0	0	0	2	4	30
25	2	1	7	4	15	2	0	1	0	5	1	3	0	1	7	0	48	0	0	0	0	0	0	0	3	27

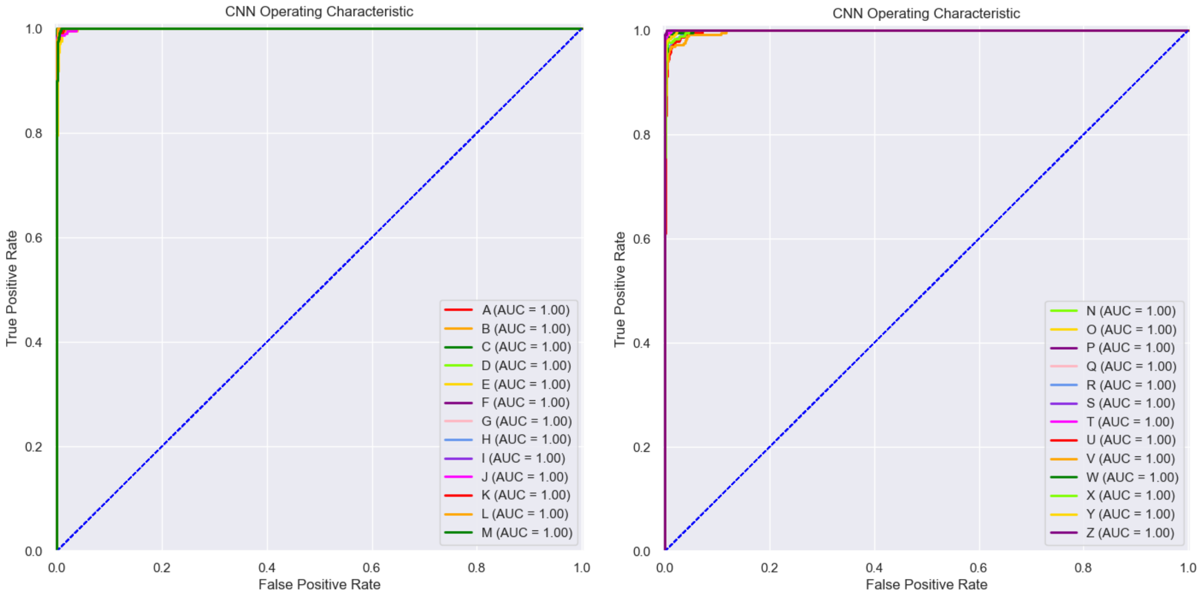
Shallow CNN Training and Testing Loss:



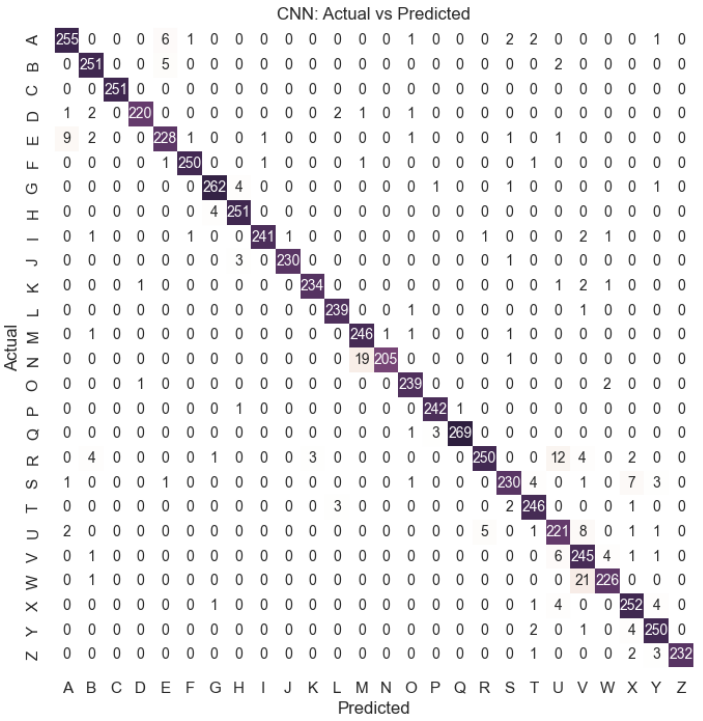
Deep CNN Training and Testing Loss:



Deep CNN ROC Curve:



Deep CNN Confusion Matrix:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	255	0	0	0	6	1	0	0	0	0	0	0	0	0	1	0	0	0	2	2	0	0	0	0	1	0
B	0	251	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
C	0	0	251	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	1	2	0	220	0	0	0	0	0	0	0	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0
E	9	2	0	0	228	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
F	0	0	0	0	1	250	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
G	0	0	0	0	0	0	262	4	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0
H	0	0	0	0	0	0	4	251	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	1	0	0	0	1	0	0	241	1	0	0	0	0	0	0	0	1	0	0	0	2	1	0	0	0
J	0	0	0	0	0	0	0	3	0	230	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
K	0	0	0	1	0	0	0	0	0	0	234	0	0	0	0	0	0	0	0	0	1	2	1	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	239	0	0	1	0	0	0	0	0	0	1	0	0	0	0
M	0	1	0	0	0	0	0	0	0	0	0	0	246	1	1	0	0	0	1	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	19	205	0	0	0	0	1	0	0	0	0	0	0	0
O	0	0	0	1	0	0	0	0	0	0	0	0	0	0	239	0	0	0	0	0	0	0	2	0	0	0
P	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	242	1	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	269	0	0	0	0	0	0	0	0	0
R	0	4	0	0	0	0	1	0	0	0	3	0	0	0	0	0	0	250	0	0	12	4	0	2	0	0
S	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	230	4	0	1	0	7	3	0
T	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	2	246	0	0	0	1	0	0
U	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	1	221	8	0	1	1	0
V	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	245	4	1	1	0
W	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	226	0	0	0
X	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	252	4	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	4	250	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	3	232