Image Classification of American Sign Language

Introduction

In the United States alone, there are approximately 12 million people with some degree of hearing loss, but only about 500,000 or 1% of this population use sign language. This low percentage stems from various factors, including lack of accessibility to learning American Sign Language (ASL) due to unique environments that an individual with hearing impairment may be subjected to. As incentivization for accessibility to these resources increase, technology to cater to individuals using ASL must increase accordingly. Using "Sign Language MNIST," a dataset consisting of pixel vectors representing images of the ASL alphabet, the goal of this project aims to utilize classification models to yield the highest accuracy model. Furthermore, applications of high accuracy models implies that individuals with hearing impairments have accessible technology platforms, eliminating disparities in available resources.

In order to build high accuracy classification-models, logistic regression, a support vector machine (SVM), and a convolutional neural network (CNN) were tested. Hypotheses were made regarding the performances of the respective models based on the following stipulations: support vector machines generally perform well on unstructured data such as images as opposed to logistic regression, and that CNN would perform the best due to its ability to model image data.

Preprocessing

Prior to modeling, a variety of transformations on the given dataset were conducted. For instance, separating the labels and the data (the vector of pixels), reshaping the labels into 2D arrays, and normalizing the data were completed. These measures ensure that the data can be manipulated in ways that are appropriate and efficient. Furthermore, since the dataset obtained consisted of 784 pixel features per sample (or per image), associated with the respective label of the image, minimal additional preprocessing steps were required.

Logistic Regression

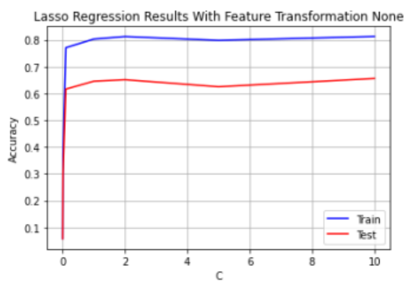
The first model tested was logistic regression, in which the data was transformed with three variations: no transformation, squared transformation, and cubed transformation (every pixel value in the vector of pixels for each example was transformed in this way). After performing logistic regression on the transformed features, the accuracy was measured based on the Sklearn logistic regression score function, which measures the mean accuracy of the data with respect to the labels. The cubed transformation yielded the highest testing accuracy of 67.7, though the difference was minimal.

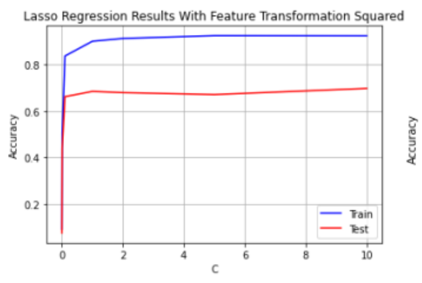
Table 1: Training and Test Accuracy with Transformed Features

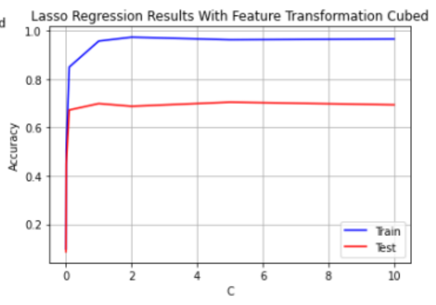
Transformation	Training Accuracy (%)	Testing Accuracy (%)
None	95.6	65.6
Squared	99.8	67.7
Cubed	100	67.6

Afterwards, in order to prevent overfitting, while also increasing accuracy, the goal was to find the optimal tuning parameter C to implement various regularization techniques such as Lasso regularization, which adds the absolute value of the coefficient's magnitude as a penalty (L1 penalty), Ridge regularization, which adds the squared value of the coefficient's magnitude as a penalty (L2 penalty), and finally Elastic-Net regularization which combines both L1 and L2 penalties.

When testing various values of C, ranging from C=.001 to C=10 on each of the transformed sets of data (no transformation, squared, cubed), training accuracy and testing accuracy were recorded and illustrated below.

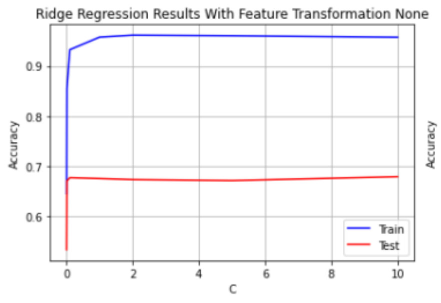


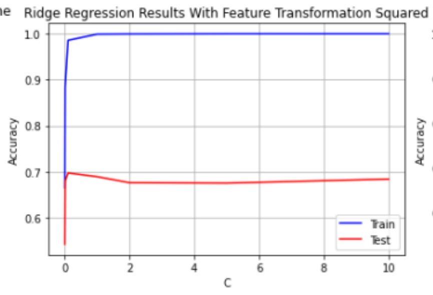


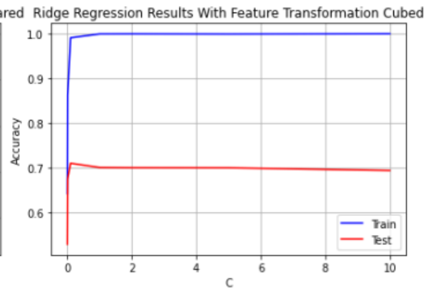


The results reveal that the cubed transformation features with lasso regularization yield the highest accuracy.

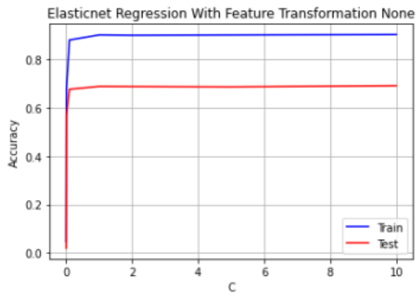
Similarly, ridge regularization was implemented on each of the results with squared, cubed, and no transformations.

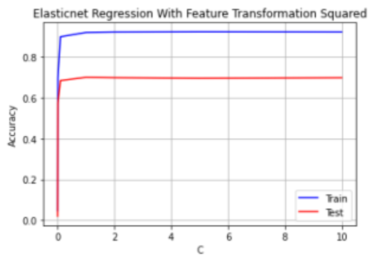


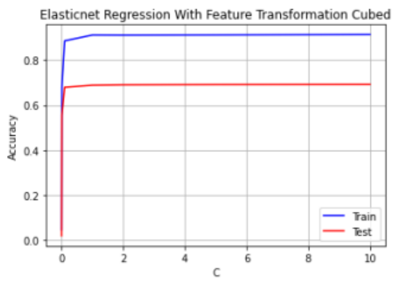




Similarly, finding the best parameter C for Elastic-net regularization was determined.

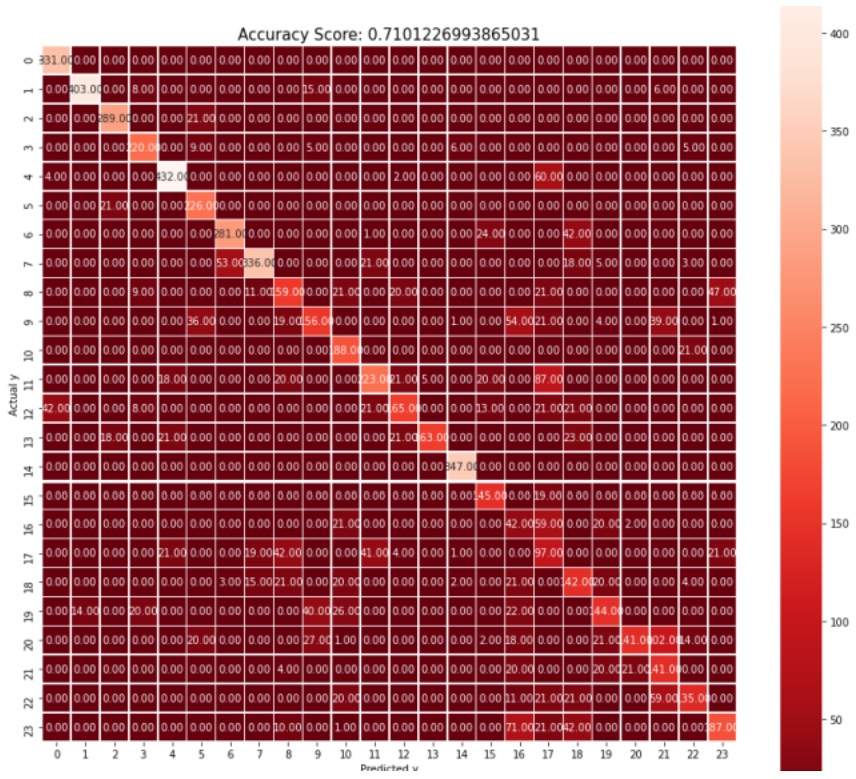






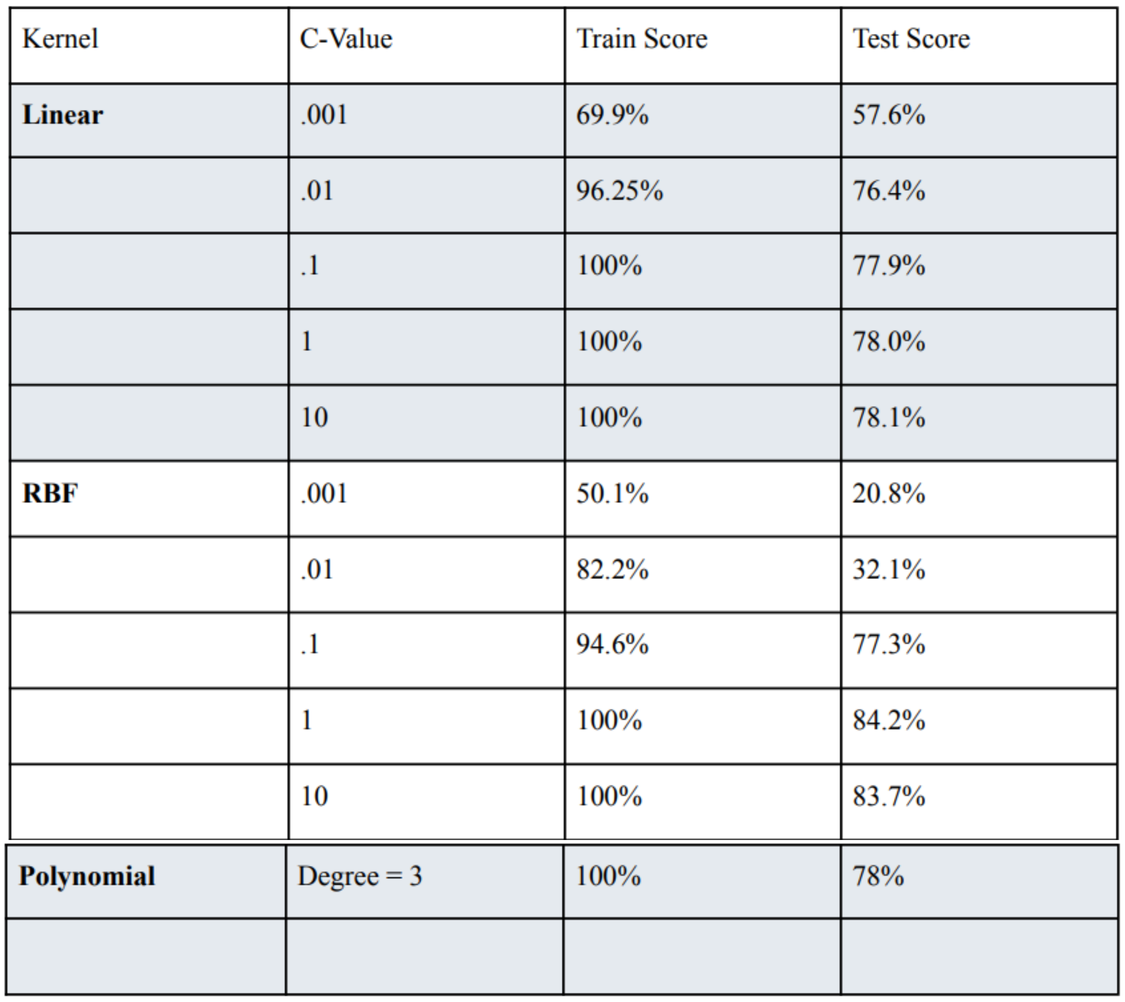
After testing each transformation against the regularization technique and optimal parameter C, Cubed feature transformations, coupled with ridge regularization, and a C-value of 0.1 was optimal.

Using this model, a confusion matrix was created to depict how the test data performed in terms of the predicted label versus the given label. The final accuracy score was approximately 71% of image labels that were predicted correctly.



Support Vector Machine

SVM Modeling was conducted with varying kernel functions: linear, RBF, and polynomial with varying C-values, 0.001, 0.01, 0.1, 0.5. The train score, or the mean accuracy between the predicted training data labels and the training data labels, as well as the test score, the mean accuracy between the predicted test data labels, and true test data labels.



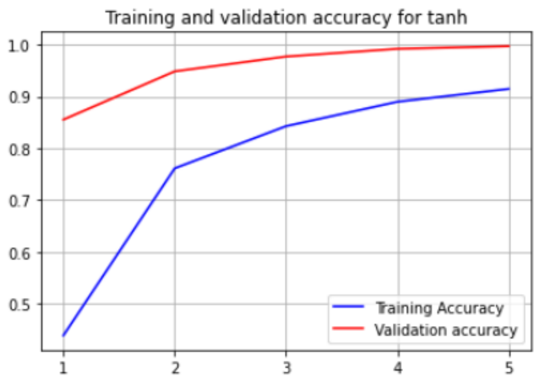
Kernel	C-Value	Train Score	Test Score
Linear	.001	69.9%	57.6%
	.01	96.25%	76.4%
	.1	100%	77.9%
	1	100%	78.0%
	10	100%	78.1%
RBF	.001	50.1%	20.8%
	.01	82.2%	32.1%
	.1	94.6%	77.3%
	1	100%	84.2%
	10	100%	83.7%
Polynomial	Degree = 3	100%	78%

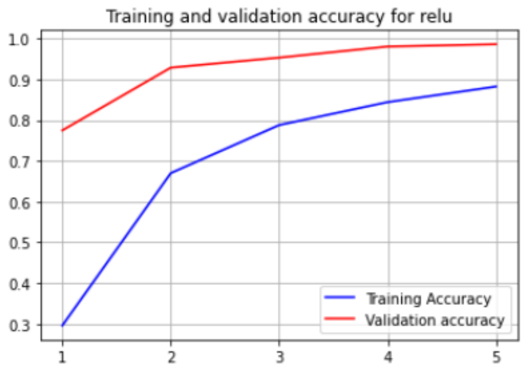
The table reveals that the RBF kernel, with a C-value of 1 performed the best on the test data. The table also reveals the effects of underfitting and overfitting by tuning the hyperparameter C to be more / less "soft" or "flexible." In addition to trying various SVM parameters - additional attempts were made to find optimal parameters using K cross-fold validations.

Convolutional Neural Networks

The final model tested, CNNs, examined how various activation functions and L2 regularization terms could impact the training and test accuracy, as well as the training and test loss of the model.

Initially, the training/test accuracy/loss was measured with three activation functions: tanh,, RELU, and linear and a 3 layer model.





The validation accuracy was recorded for each activation type.

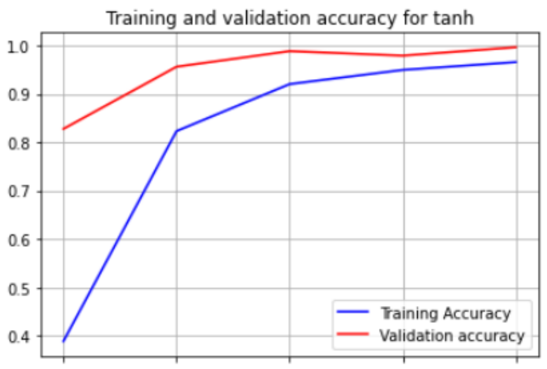
Activation Type	Validation Accuracy (%)
Tanh	99.7%
Linear	98.6%
Relu	98.9%

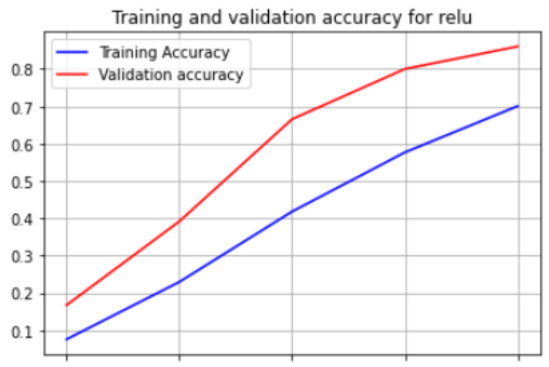
Similarly, the same 3 layer model was run after L2 regularization was performed, and the training/validation accuracy was measured for each.

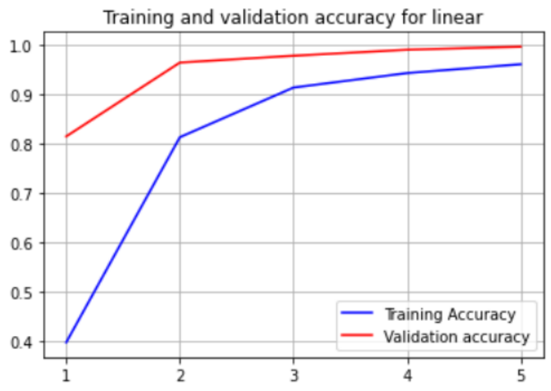
3 Layer Model Accuracy measured from activation function vs. parameter

	.0001	.001	0.1
Tanh	99.5	99.5	92.8
Linear	98.7	99.8	75.9
RELU	98.1	98.3	75.9

Additionally, a 5-Layer Model was tested. Initially, batch normalization and dropout layers were added, but replaced with two additional layers. The goal of adding layers is to reduce possible overfitting, while simultaneously increasing the performance of the CNN.







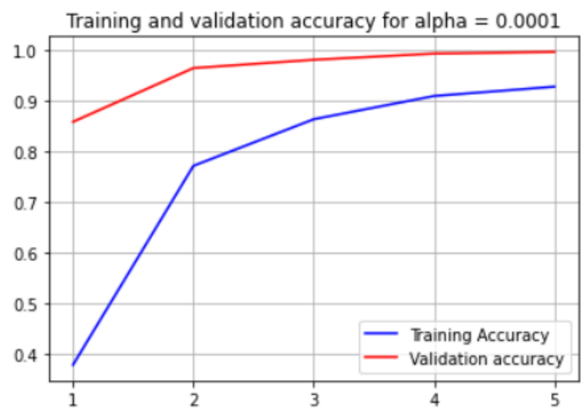
Activation Type	Validation Accuracy (%)
Tanh	99.6%
Linear	99.7%
Relu	86.0%

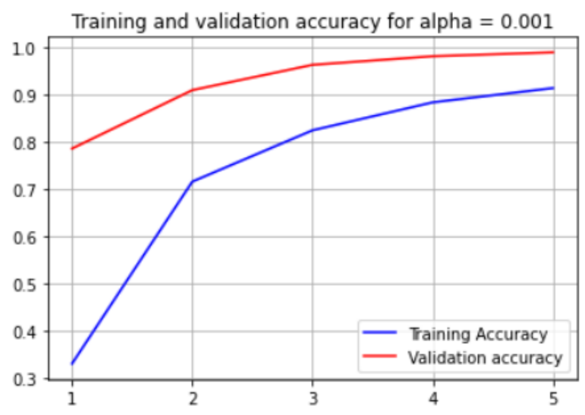
L2 regularization terms were added to the 5 layer model.

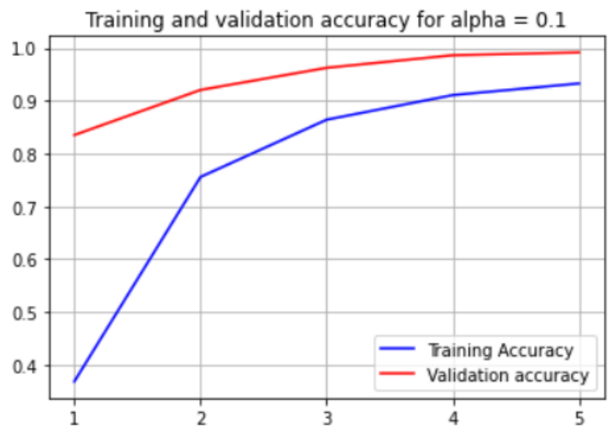
5 Layer Model Accuracy (%) measured from activation function vs. L2 regulation values

	.0001	.001	0.1
Tanh	99.5	99.3	4.5
Linear	99.1	98.9	93.6
RELU	85.4	84.3	4.2

Finally, the Leaky Relu activation function was tested with varying alphas: 0.0001, .001, and 0.1. Leaky Relu, sends small slopes rather than zero to use for negative values, as opposed to the Relu activation function.







Leaky RELU Validation accuracy with varying alphas

Alpha	Validation Accuracy (%)
.0001	99.6
.001	99.8
.1	99.8

Conclusion

Ultimately, through modeling the effects of parameters, as well as the effects of overfitting and underfitting were shown. For instance, modifying the regularization term from .001 to .01 could decrease the accuracy from 98% to 4% as seen in the CNN 5 layer model with L2 regularization. From previous research, it is known that when the hyperparameter, C, is large it is expected that the model will try to overfit, yielding high, or even perfect training accuracy scores. On the other hand, when C is extremely small, the model will underfit since there is too much flexibility in the classification.

Additionally, when manipulating the data in various ways - using only a portion of the data in order to decrease runtime showed how decreasing the size of the dataset can lead to underrepresentation of the data.