Identifying ASL Fingerspelling Images with Machine Learning

Kyle Donoghue and Nick Russert

***Abstract*—This paper presents a machine learning approach to identifying American Sign Language (ASL) fingerspelling images. The goal of this project is to develop a model that can accurately identify the letters of the alphabet in ASL fingerspelling images. Two different machine learning models are trained and evaluated on a dataset of ASL fingerspelling images: a convolutional neural network (CNN) and a support vector machine (SVM). The performance of the models is evaluated using standard classification metrics such as accuracy and F1 score. The results of the CNN and SVM models are compared, and the strengths and weaknesses of each model are discussed. For feature extraction for image data, the SVM model uses the Histogram of Gradients (HOG) technique. The paper also discusses the major challenges faced during the project and the solutions that were implemented. The results of the project show that both the CNN and SVM models are able to achieve high accuracy on the ASL fingerspelling dataset, with the SVM model achieving 100% accuracy and the CNN model achieving 99.5% accuracy. The results of the project demonstrate the effectiveness of machine learning models for identifying ASL fingerspelling images and provide insights into the strengths and weaknesses of different models for this task. Additionally, dimensionality reduction using principal component analysis (PCA) is explored for the SVM model, showing that the model can achieve near 100% accuracy with an 86% reduction in feature vector length.**

Source code for this project can be found at for the CNN model and [*SVM Drive*] for the SVM model. The source codes are presented as .zip files.

I. Introduction

THIS paper presents a machine learning approach to identifying American Sign Language (ASL) fingerspelling images. The goal of this project is to develop a model that can accurately identify the letters of the alphabet in ASL fingerspelling images. In this project, two different machine learning models are trained and evaluated on a dataset of ASL fingerspelling images: a convolutional neural network (CNN) and a support vector machine (SVM). The details of the models are discussed later in the paper. The performance of the models is evaluated using standard classification metrics such as accuracy and F1 score. The results of the CNN and SVM models are compared, and the strengths and weaknesses of each model are discussed. The paper is organized as follows: Section II provides an overview of the task, dataset, and models used in the project. Section III discusses the major challenges faced during the project and the solutions that were implemented. Section IV presents the experimental results and analysis. Finally, Section V concludes the paper and discusses potential future work.

II. Project Overview

A. Task Description

The task of identifying ASL fingerspelling images involves classifying images of the fingerspelling alphabet in American Sign Language. The ASL fingerspelling alphabet consists of 26 letters, each of which is represented by a specific handshape. Figure 1 shows an example of the ASL fingerspelling alphabet. The goal of the machine learning models is to use the images of the fingerspelling alphabet to predict the corresponding letter. The task is a discrete classification problem, where the input is an image of a hand making a specific handshape, and the output is the corresponding letter of the alphabet.

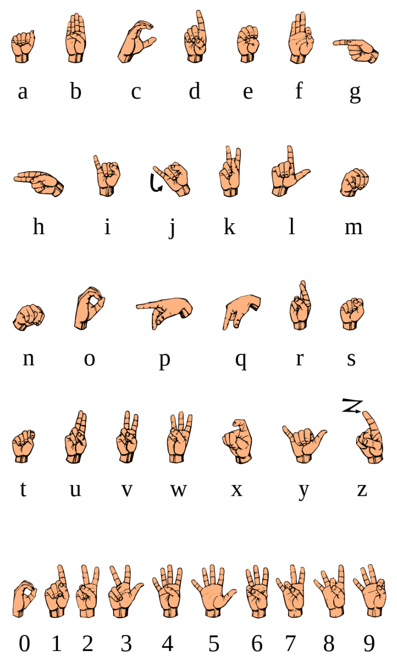

Fig. 1. The ASL fingerspelling alphabet. Each letter of the alphabet is represented by a specific handshape. The goal of the machine learning models is to use images of the fingerspelling alphabet to predict the corresponding letter. *Image Source*.

B. Dataset

For this task, we are using an open-source dataset procured from Kaggle [1]. The dataset consists of 87,000 images of the ASL fingerspelling alphabet, with 3,000 images per letter. This dataset not only includes the standard 26, but also includes images for 'space,' 'delete,' and 'nothing.' Each of the 29 labels have 3,000 RGB images of size 200x200 pixels. The dataset is split into training and testing sets. The size of the training set is variable and is adjusted for each model to see how the size of the training set affects the model's performance. The testing set is fixed at 21,750 images, with 750 images per label. The training and testing sets are balanced, with an equal number of images for each label. The dataset is preprocessed by converting the images to grayscale and normalizing pixel intensities to the range [0, 1].

C. Models

Two different machine learning models are used for this task: a convolutional neural network (CNN) and a support vector machine (SVM). The CNN is a deep learning model that is well-suited for image classification tasks. The following sections provide an overview of the CNN and SVM models used in this project.

1) Convolutional Neural Network (CNN): The Convolutional Neural Network (CNN) is a powerful deep learning model widely used for image classification tasks. Unlike other traditional machine learning models, CNNs excel at automatically learning intricate patterns and features from raw pixel data without the need for additional feature extractions and engineering.

The CNN operates by passing input images through a series of convolutional layers, pooling layers, and fully connected layers. The layers work to extract hierarchical features from the input images in order to classify them into their respective groups. Activation functions, such as ReLu, introduce non-linearity to the model, allowing for it to capture more complex relationships within the input images.

For this project, the images undergo preprocessing and resizing from 200x200 pixels to 100x100 pixels (this is due to memory space mentioned in Section III). Unlike the feature extraction approach used in the SVM model, CNNs allow for directly ingesting raw pixel values of images. This removes the need for explicit feature extraction such as the Histogram of Gradients (HOG) technique. As a result, the model automatically learns relevant features through the convolution layers, capturing differences within the images.

The CNN model is trained using the TensorFlow/Keras library in Python. During the training of the model, parameters are fine-tuned using back-propagation and gradient descent techniques. Hyperparameters, including the learning rate and batch size, are adjusted to find and achieve optimal performance on the training data set. Further, dropout layers are utilized to prevent overfitting and improve the model's generalization to unseen data.

After training the model is evaluated on a separate testing set to assess its performance which includes accuracy, precision, recall, and F1-score. This information allows us to view the model's ability to correctly classify the images into the 29 separate classes.

2) Support Vector Machine (SVM): The SVM is a supervised machine learning model that is used for classification tasks. The SVM works by finding the hyperplane that best separates the data into different classes using a specified kernel function. In this project, both a polynomial kernel and a radial basis function (RBF) kernel are used and compared.

The input of the SVM model involves further preprocessing of the images. The Histogram of Gradients (HOG) technique is used to extract a feature vector from each image. The length of this feature vector is determined by the following equation:

$$\mathrm{L} = \left(\frac{\mathrm{img}_{\mathrm{size}} - \mathrm{block}_{\mathrm{size}}}{\mathrm{block}_{\mathrm{stride}}} + 1\right)^2 \times N_{\mathrm{bins}} \times \left(\frac{\mathrm{block}_{\mathrm{size}}}{\mathrm{cell}_{\mathrm{size}}}\right)^2 . \quad (1)$$

For this model, block size is set to 80 pixels, block stride is set to 40 pixels, cell size is set to 40 pixels, and the number of bins is set to 9. Such large pixel values are used because the images in the dataset are quite coarse in edge size, meaning that there are not many fine details needed to capture to differentiate between the different letters. These parameters lead to a feature length of 576. The calculated feature vector is then used as input to the SVM model. Using the HOG technique allows the SVM model to capture the shape and orientation of the edges in the images, while also considerably reducing the dimensionality of the input data (from 40,000 pixels to 576 features).

The SVM model is trained using the scikit-learn library in Python. The hyperparameters of the SVM model, such as the kernel type and regularization parameter, are tuned using grid search with 5-fold cross-validation on the training set. The best hyperparameters are selected based on the average accuracy across the 5 folds. Besides the kernel choice, the other hyperparameters varied in the grid search are C and γ. The best model from the grid search is then evaluated on the testing set to obtain the final performance metrics.

III. Major Challenges and Solutions

A. CNN Feature Length

One of the major challenges encountered during the development of the CNN model was memory constrained during processing the images. Due to the large data set, attempting to allocate memory for the array of images resulted in the following error message, "*unable to allocate 9.72 GiB for an array with shape (87000, 200, 200, 3)*". This error indicates that the system was unable to allocate the required memory to accommodate the original size of the image data.

To work around the memory issue, the implemented solution was to resize the input images to a smaller dimension. By reducing the images from 200x200 pixels to 100x100 pixels, the chunk of memory needed to process the images was significantly reduced. This allowed the CNN model to process the images without throwing a memory allocation error.

While resizing the images addressed the memory problem and allowed for the CNN model to run with no errors, it is imperative to mention the potential issues that arise with resizing the images. Reducing the number of pixels within the images may result in a loss of spatial information and detail. This could potentially affect the model's ability to accurately capture fine details and information contained within the images. For the purpose of this project, however, the impact of image resizing on model performance was deemed acceptable, considering it was essential to fully run and train the model

B. SVM Hyperparameter Tuning

With the SVM model, the hyperparameters that need to be tuned are the kernel type, the regularization parameter C, and the kernel coefficient γ. The challenge with tuning these hyperparameters is that the search space is large, and the optimal hyperparameters can vary depending on the dataset and the model. To address this challenge, a cross-validated grid search is used to find the best hyperparameters for the SVM model. The grid search is performed using 5-fold cross-validation on the training set, and the hyperparameters that yield the highest average accuracy across the 5 folds are selected as the best hyperparameters. This approach helps to ensure that the hyperparameters are tuned effectively and that the model is not overfitting to the training set. Specifically, the grid search is performed over the following hyperparameters: kernel type (polynomial and RBF), C (0.1, 1, 10, 100), and γ (0.0001, 0.001, 0.1, 1).

Because SVM is a computationally expensive model, the grid search is performed using a subset of the training set to reduce the computational cost. The grid search is performed with 500 images per label, which results in a total of 14,500 images in the training set. Remember, the full training set has 65,250 images, so this is a significant reduction in the number of images used for the grid search. The best hyperparameters found using the subset of the training set are then used to train the SVM model on the full training set. This approach allows us to find the best hyperparameters for the SVM model while also managing the computational cost of the grid search.

C. Similar Handshapes

One of the major challenges faced during the project was the similarity between the handshapes of the ASL fingerspelling alphabet. Many of the handshapes are visually similar, making it difficult for the models to differentiate between them. For example, the letters ‘A,’ ‘M,’ and ‘N’ have similar handshapes, with only subtle differences in finger positions. This similarity in handshapes can lead to misclassifications by the models, as they may have difficulty distinguishing between the different letters. Additionally, we have seen with our models that the letters ‘S,’ ‘U,’ and ‘X’ are difficult to classify, as they have similar handshapes that are challenging to differentiate.

IV. Experiments

A. Evaluation Metrics

The performance of the CNN and SVM models are primarily evaluated using accuracy, but F1 score is also used to provide a more detailed analysis of how each model performs on each class. The accuracy is calculated as the proportion of correctly classified images to the total number of images in the testing set. The F1 score is calculated as the harmonic mean of precision and recall, where precision is the proportion of true positive predictions to the total number of positive predictions, and recall is the proportion of true positive predictions to the total number of actual positive instances. The F1 score provides a more balanced measure of the model’s performance, especially in cases where the classes are imbalanced, which might be the case with similar handshapes in the ASL fingerspelling alphabet.

V. Results

A. Hyperparameter Selection

For the CNN model, hyperparameters were chosen through a series of experiments to optimize the performance of the model. A batch size of 8 was chosen as a trade-off between training speed and model convergence. We set the number of epochs to 10. This number determines how many times the model iterates over the entire training set during the training of the model. We also trained the model at 5 and 20 epochs and found that at 5 epochs the accuracy was lower than desired and at 20 epochs the model converged before finishing, indicating that 20 was too many epochs. For the optimizer and learning rate, we used Adam. This is an adaptive learning rate optimization algorithm that adjusts the learning rate based on the gradients magnitudes. Figure 2 below shows the increase in training and validation accuracy over the training epochs.

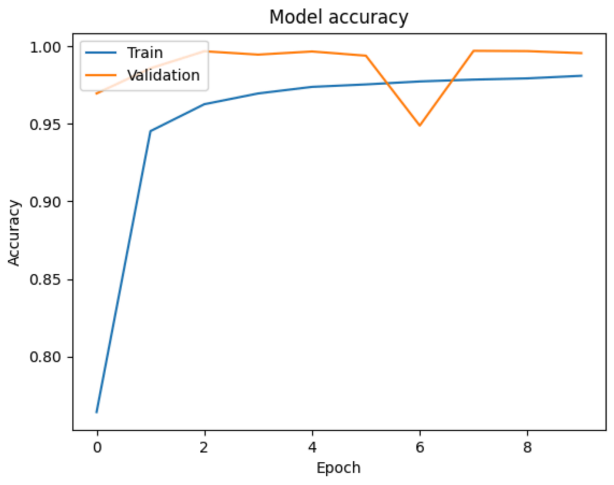

Fig. 2. Graph illustrating the performance of the CNN model over multiple epochs during the training. Each epoch represents a complete pass through the entire training set. Here the Y axis is accuracy and X is the number of epochs. Ultimately this graph shows the model achieving higher accuracy with each epoch. Note that there was a bad validation score on epoch 6, hence the dip.

For the SVM model, the hyperparameters were chosen with a 5-fold cross-validated grid search. The grid search was performed over the following hyperparameters: kernel type (polynomial and RBF), C (0.1, 1, 10, 100), and γ (0.0001, 0.001, 0.1, 1). Figure 3 shows a 3D surface plot of the average accuracy across the 5 folds for different values of C and γ for the polynomial and RBF kernels. The best hyperparameters were found to be the polynomial kernel with $C = .1$ and $\gamma = .1$. It should be noted that while the polynomial kernel was chosen as the best kernel for this model, the polynomial kernel also demonstrated more stability in the grid search, as the RBF kernel had steeper peaks and valleys in the accuracy surface plot. This suggests that the polynomial kernel may be more robust to changes in the hyperparameters.

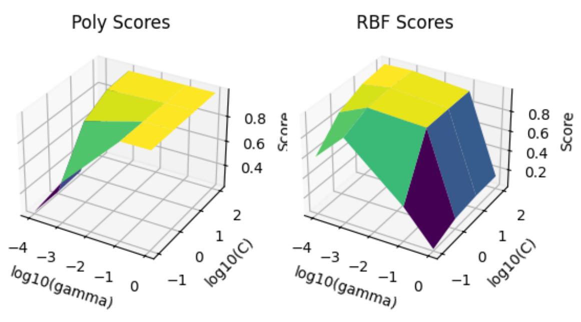

Fig. 3. 3D surface plots of the average accuracy across the 5 folds for different values of C and γ for the polynomial and RBF kernels.

B. Best Results

The best CNN model was found to be the one with 4 convolutional layers, 2 fully connected layers, and ranging from 32 up to 128 filters in each convolution later. The model was trained using the adaptive learning rate optimizer Adam with a batch size of 8. Using the cross-validated grid search, the best SVM model was found to be the polynomial kernel with $C = .1$ and $\gamma = .1$. Table I shows the best results obtained by both the CNN and SVM models. This table shows both the accuracy and minimum F1 score obtained by each model. The minimum F1 score is used to show the worst performance of the model on any given class.

TABLE I
BEST RESULTS OF CNN AND SVM MODELS

Model	Accuracy	Minimum F1 Score (Label)
CNN	99.5%	98.7% (S)
SVM	100.0%	100.0% (N/A)

In addition to these tabular results, we can also present the confusion matrices for both the CNN and SVM models. These matrices show how well the models are able to classify each class. The confusion matrices for the CNN and SVM models are shown in Figures 4 and 5.

C. Varying Training Size

With both the CNN and SVM, we can see how the size of the training set affects the model's performance. The training set size is varied from 10 images per label to 2250 images per label. The evaluation metric used for this test is the validation score of the model for each training set size. The results of the CNN and SVM models with varying training set sizes are shown in Figures 6 and 7.

To give more context, we can also show the confusion matrices for a lower training size to see how the confusion matrix changes as the training set size decreases. This can give insight into how the model is performing with less data.

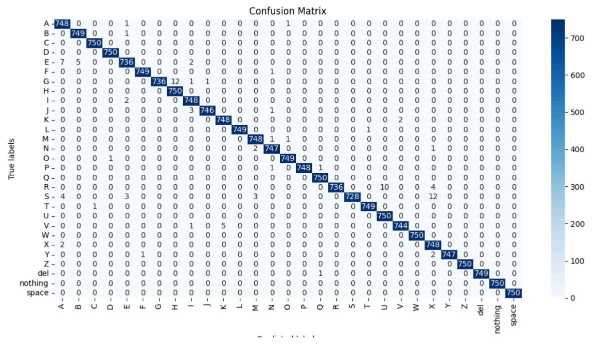

Fig. 4. Confusion matrix for the CNN model with 2250 images per label and optimal hyperparameters. This matrix shows the number of images that were correctly classified and misclassified for each class. For this specific model, the confusion matrix is a diagonal matrix with most of the values along the diagonal being close to 750. We can see the worst performing label for this model was S with 728/750 being correctly classified.

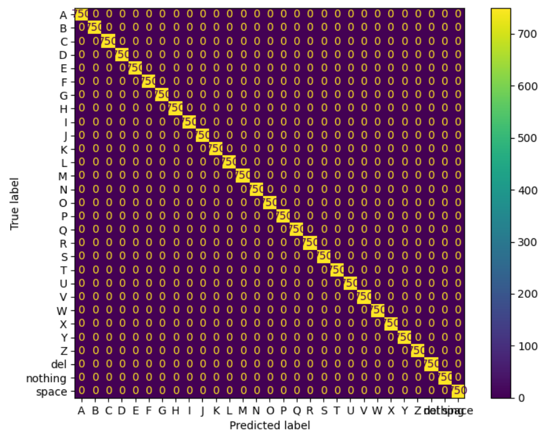

Fig. 5. Confusion matrix for the SVM model with 2250 images per label and optimal hyperparameters. This matrix shows the number of images that were correctly classified and misclassified for each class. For this specific model, the confusion matrix is a diagonal matrix with all the values on the diagonal being 750, which indicates that the model was able to correctly classify all the test images.

The confusion matrices for the CNN and SVM models with 250 images per label are shown in Figures 8 and 9.

D. Varying Test Image SNR

To evaluate the robustness of the CNN and SVM models to noise, we add Gaussian noise to the testing images at varying signal-to-noise ratios (SNRs). The SNR is defined as the ratio of the signal power to the noise power, and is given by the equation:

$$\text{SNR (dB)} = 10 \log_{10}\left(\frac{P_X}{P_N}\right), \tag{2}$$

where P_X is the average power of the signal and P_N is the average power of the noise. The SNR is varied from -10 dB to 40 dB, where 0 dB represents a high level of noise and 30 dB represents a low level of noise. This evaluation also shows the quality of the fit of the model, as if the model is overfitting, it will not perform well on the noisy images.

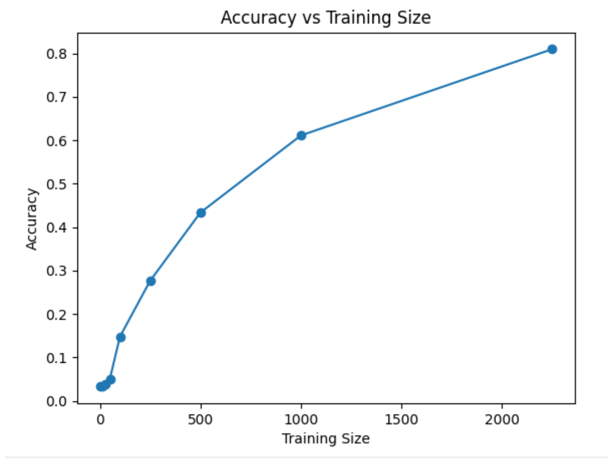

Fig. 6. Graph for accuracy vs training size for the CNN model. With increased training size comes increased accuracy.

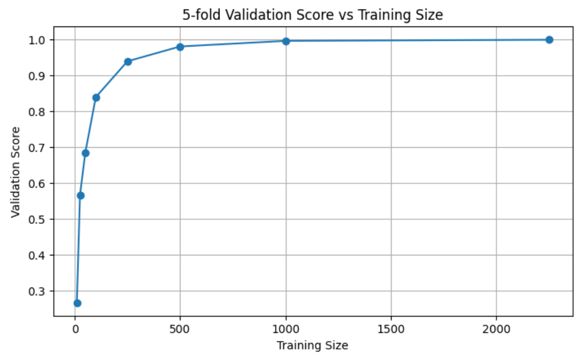

Fig. 7. Performance of the SVM model with varying training set sizes and the same hyperparameters. This plot shows the accuracy of the SVM model as the training set size is varied from 10 images per label to 2250 images per label. After 1000 images per label, the model's performance plateaus, suggesting that additional data does not significantly improve the model's performance.

For the CNN and SVM models, they were originally trained using the unaltered images. The testing images were then altered to have varying levels of noise. The models were then evaluated on these noisy images to see how the model's performance changes with increasing noise levels. The results of the models with varying SNRs are shown in Figures 10 and 11. These images also show the performance of the models when trained with images under noise. This was done to see if the models could generalize better to noisy images if they were trained with noisy images. Section VI will discuss the results of this evaluation.

E. SVM PCA

As a final experiment, we can also show the results of the SVM model when using principal component analysis (PCA) to reduce the dimensionality of the feature vectors. The same HOG feature vectors are extracted from the images, but are then passed through a PCA transformation to reduce the dimensionality of the feature vectors. The total variance explained by the principal components is calculated, and the number of principal components is chosen to capture a certain percentage of the variance. Here, we choose to capture 70%, 80%, and 90% of the variance in the data, and the SVM model is trained and evaluated using the reduced feature vectors. These results of the SVM model with PCA to reduce dimensionality are shown in Table II. Additionally, Figure 12 shows the explained variance ratio of the principal components. This plot shows how much of the variance in the data is explained by each principal component, and can help determine how many principal components are needed to capture most of the variance in the data. Remember that, originally, the feature vector length was 576.

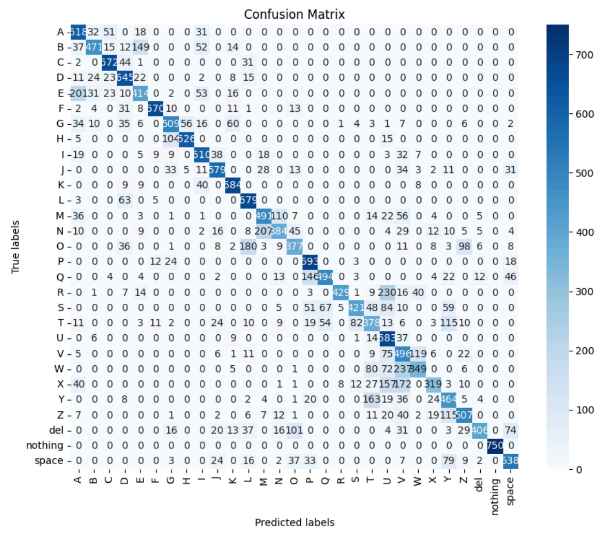

Fig. 8. Confusion matrix for the CNN model, trained with 250 images per label and optimal hyperparameters. This matrix shows the number of images that were correctly classified and misclassified for each class. For this specific model, the model was able to correctly classify some test images, but had much more difficulty than when trained with the full data set of 2250 training images. The only case this model was able to identify 100% of the time was 'nothing'. Some common misclassifications are 'N,' 'M,' 'X,' and 'Y.' Note the images for this training and testing are still 100 x 100 pixels.

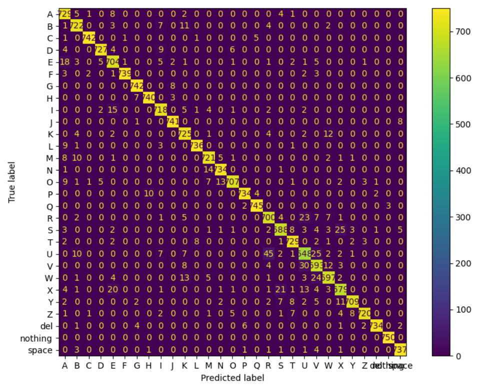

Fig. 9. Confusion matrix for the SVM model with 250 images per label and optimal hyperparameters. This matrix shows the number of images that were correctly classified and misclassified for each class. For this specific model, the model was able to correctly classify most of the test images, but had some difficulty with certain classes, such as 'S,' 'U,' and 'X' where these classes were correctly classified less than 680 times.

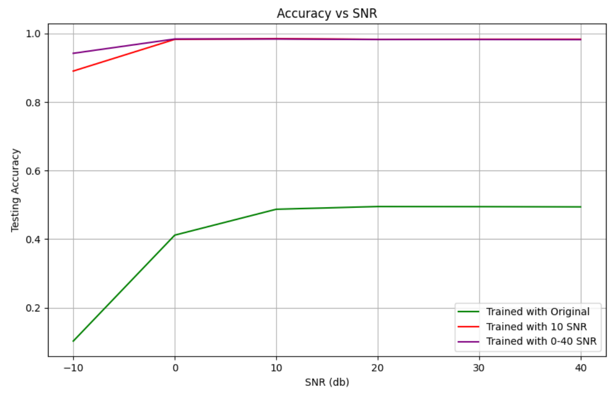

Fig. 10. Performance of the CNN model with varying SNRs. This plot shows the accuracy of 3 different SVM models: one trained with unaltered images, one trained with images under 10 dB of SNR, and one trained with images under 0-40 dB of varying SNR. The models are evaluated on images with SNRs ranging from -10 dB to 40 dB.

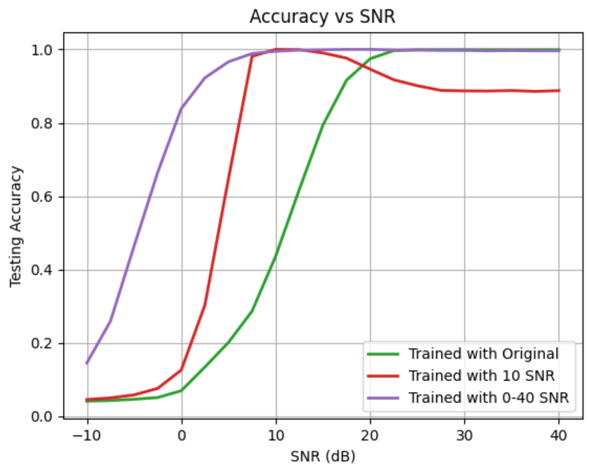

Fig. 11. Performance of the SVM model with varying SNRs. This plot shows the accuracy of 3 different SVM models: one trained with unaltered images, one trained with images under 10 dB of SNR, and one trained with images under 0-40 dB of varying SNR. The models are evaluated on images with SNRs ranging from -10 dB to 40 dB.

F. Prediction Failures

In order to gain insight into the performance of the CNN model, we conducted a visualization of instances where the model misclassified the images. This allowed us to identify potential areas of improvement and understand some challenges the model was facing. Some predicted reasons for the misclassifications include the complexity of the gestures. Many of the hand signs are very similar visually making it challenging for the model to differentiate between them. As seen in Figures 13 and 14, subtle variations in finger positions may be misclassified due to their close resemblance. On top of that, since the images were reduced to 100 x 100 this presented further challenges as defining details may have been lost.

TABLE II
SVM Model with PCA Results

Var. Captured	# Components	Acc.	Min. F1 Score (Label)
60%	12	98.41%	94.63% (U)
70%	20	99.45%	97.00% (U)
80%	36	99.98%	99.87% (V)
90%	80	99.99%	99.86% (R)

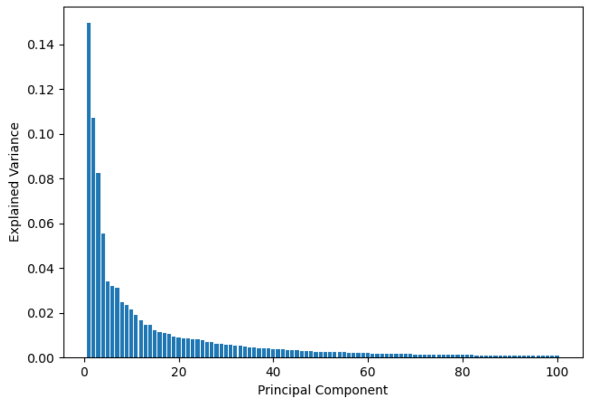

Fig. 12. Plot showing the explained variance ratio of the first 100 principal components.

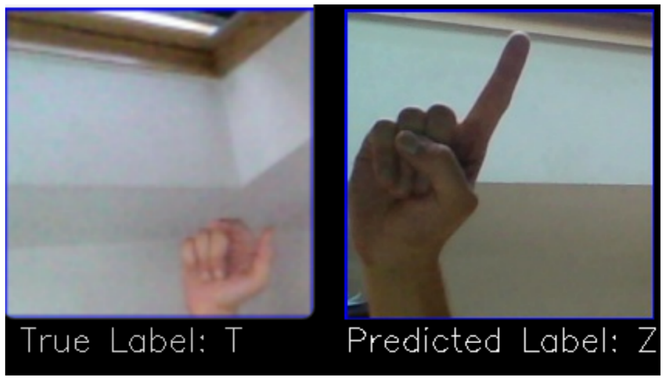

Fig. 13. Example of the CNN model predicting an ‘T’ as an ‘Z’.

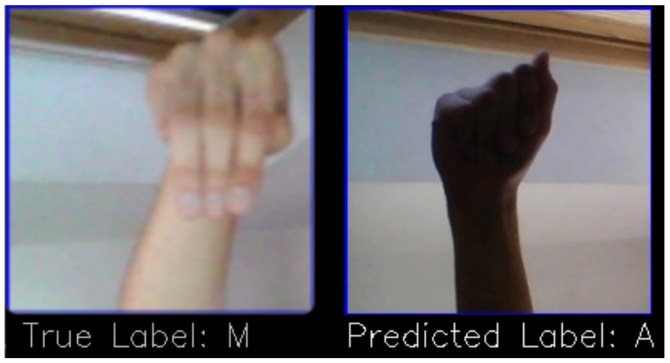

Fig. 14. Example of the CNN model predicting an ‘M’ as an ‘A’.

VI. Discussion

A. Training Size Impact

For the CNN model, we see the expected result of increased training size resulting in increased accuracy of the model. This is the expected output as more training samples means the model has more information to learn from. We see that the model achieves its max accuracy at 2250 training images per label. We see that the full training data is necessary for the model to achieve its max potential. The model doesn't plateau within these amounts of testing data sizing meaning that with even more data, the model might become even more accurate. It is important to acknowledge that for this figure, the model was trained with the amount of training data on the x-axis but tested on the full data set of 750 images. This suggested that when it is trained with fewer images, the model does not generalize well to the remaining unseen images because it has a lower accuracy than desired.

For the SVM model, we can see the expected trend that as the training set size increases, the model's performance improves. This is because the model has more data to learn from, and can better capture the underlying patterns in the data. In this case, though, we can see that the performance of the SVM model plateaus after a certain training set size. Here, we achieve 99.77% accuracy at 1000 images per label, only a 0.27% decrease from the 2250 images per label. This suggests that the SVM model is able to learn the underlying patterns in the data quite well, and that additional data past this point does not significantly improve the model's performance. Actually, the SVM model is able to achieve over 90% accuracy with only 250 images per label, which is quite impressive. This suggests that the SVM model is able to generalize well to unseen data, even with a relatively small training set size. Though, we do see the SVM's performance taper off quite quickly once the training set size is reduced to under 100 images per label, where the model's accuracy drops to as low as 26.55% for 10 images per label. This accuracy is only slightly better than random guessing, which suggests that 10 images per label is not enough data for the SVM model to be able to effectively predict the labels. In this test procedure, 5-fold cross validation scores are used to be able to consistently use the same hyperparameters for each training set size, since the optimal hyperparameters can change with the size of the training set.

B. SNR Performance

For the CNN model, interesting patterns emerge from examining the results depicted in figure 9. We can see some distinct behaviors across the different training methods in response to the various SNR levels. From the original model trained with no noise, the accuracy maxes out around 50% regardless of the SNR levels. We found this very interesting as the model continues to perform poorly even as the images begin to have less noise. We hypothesize this to be the behavior because the CNN model classifies the images at a pixel level and even the slightest pixel alteration may throw the model off. This indicates that the model generalized relatively poorly to noisy conditions and may be over fitting to our specific data set. To get around this we explored training the data with a fixed SNR of 10 dB and various SNR ranging from 0-40 and saw a drastic improvement. The models after training on images with SNR added started at around 90 and 95% accuracy, respectively, and stayed around the same for the remaining SNR. This suggests that exposing the model to a diverse range of noise during the training enhances its ability to generalize to unseen data. These findings underscore the importance of training the CNN with some variability to help it generalize to unseen data and help prevent overfitting.

For the SVM model, we see some interesting characteristics when referencing Figure 11. In this figure, we see that with unaltered images, the SVM model only has an accuracy close to 1 for SNR levels above 20 dB. As the SNR decreases, the model's accuracy decreases as well. To address this, the SVM was trained using images under 10 dB of SNR. In this case, the SVM model performed better at slightly lower SNR levels, but actually reduced accuracy at higher SNR levels. This suggests that the SVM model is overfitting to the noisy images, and is not generalizing well to the unaltered images. This was addressed by training the SVM model with images that had varying SNR from 0 to 40 dB. The additive noise power was a uniform random variable to keep the classes balanced. When this model was evaluated, it achieved near 1 accuracy at even lower SNR levels than the 10 dB model, and maintained high accuracy as SNR increased, as well. This suggests that using a more diverse training set can help the model generalize better to unseen data, even if that diversity is artificially created.

C. Failure Analysis

When looking at specific examples of failure, we can see that the CNN model is able to correctly classify most of the images, but has some difficulty with certain classes. Figures 13 and 14 show examples of images that were misclassified by the CNN model. In the first example, the model predicted an 'T' as an 'Z.' This may be because the thumb on the 'T' is slightly away from the hand, making the model predict it to be closer to a 'Z.' The T image is also further into the background of the image making it not as prominent. When comparing the T and Z we see the main difference is the index finger pointing straight or up with the pinky, ring, and middle fingers all being in the same position. In the second example, the model predicted an 'M' as an 'A.' Once again we see these ASL letters are fairly similar leading to the misclassification of the image. The main difference between these two letters are that for the 'M' the fingers protrude past the wrist, more of a downward position with the thumb tucked, whereas the 'A' is more upright with the thumb at the side of the hand. With these two examples, we can see that the similar handshapes and finger positions of the ASL letters can lead to misclassifications by the CNN model.

Another point for failure analysis is the overfitting done by the SVM when presented with images of a static SNR level. In both the noise-free and 10 dB cases, the SVM model was unable to generalize well to the unseen data. What was quite surprising was the model's inability to be able to correctly predict images with a high SNR level, when the model was trained on a lower SNR level. One would assume that if the model is trained at a lower SNR level, it would be able to generalize well to higher SNR levels, but this was not the case. This suggests that the SVM model is overfitting to the training data, and a more diverse training set was needed to improve the model's generalization to unseen data. This was implemented with the 0-40 dB SNR training set, which was able to achieve near 1 accuracy at all SNR levels above 5 dB.

Looking at the confusion matrix for the 2250 images per label SVM model, we can see that the model is able to correctly classify all the test images, which doesn't give much insight to the limitations of the SVM model. We can instead look at the confusion matrix for the 250 images per label SVM model, which shows that the model is able to correctly classify most of the images, but has some difficulty with certain classes. For example, the model has the most difficulty with the letters 'S,' 'U,' and 'X.' This suggests that these letters may have similar handshapes that are difficult for the model to differentiate with the HOG feature extraction method. This also might be due to the coarse parameter settings used for the HOG feature extraction, which may not capture the fine details needed to differentiate between these letters. This is a limitation of the SVM model, as it relies on the HOG feature extraction method to capture the shape and orientation of the edges in the images. If the HOG parameters are not set correctly, the model may not be able to effectively differentiate between the classes.

D. PCA Dimensionality Reduction

For the SVM model, we can see that using PCA to reduce the dimensionality of the feature vectors can significantly decrease the model's computational complexity. In this case, we were able to reduce the number of features from 576 to 80 while still capturing 90% of the variance in the data and achieving near 100% accuracy. This suggests that the SVM model is able to effectively capture the underlying patterns in the data with a reduced feature vector. If accuracy tolerance is decreased to 98% the number of features can be reduced to 12. This is a significant reduction in the number of features, which can help to reduce the computational cost of the model and improve the model's performance. Using only 12 features, the SVM model's feature length is reduced by 97.9% while still achieving 98.41% accuracy. This can help to improve the model's performance and reduce the computational cost of the model, making it more efficient and scalable to larger datasets. Overall, these results show that PCA can be an effective method for reducing the dimensionality of the feature vectors and improving the performance of the SVM model on a HOG feature extraction model.

E. Model Comparison

When comparing the CNN and SVM models, we see that both models are able to achieve high accuracy on the ASL fingerspelling alphabet dataset. The CNN model achieves an accuracy of 99.5%, while the SVM model achieves an accuracy of 100.0%. This suggests that both models are able to effectively classify the images into their respective classes. The CNN's slightly lower accuracy could be due to the model's reliance on the raw pixel values of the images, which may not be as effective at capturing the shape and orientation of the edges in the images as the HOG feature extraction method used by the SVM model. Overall, both models are able to achieve high accuracy on the dataset, but the SVM model may be better suited for this task due to its ability to effectively capture the underlying patterns in the data and classify the images into their respective classes.

Additionally, we see that the SVM model is able to significantly reduce its feature vector length using PCA, while still achieving high accuracy on the dataset. The CNN model, on the other hand, is able to better operate under lower SNRs than the SVM model. This suggests that the CNN model may be better suited for tasks that require robustness to noise, while the SVM model may be better suited for tasks that require dimensionality reduction and computational efficiency. Overall, both models are able to effectively classify the images into their respective classes, but may be better suited for different tasks depending on the requirements of the task.

VII. Conclusion

In this project, we developed and evaluated two machine learning models, a Convolutional Neural Network (CNN) and a Support Vector Machine (SVM), for classifying the American Sign Language (ASL) fingerspelling alphabet. The CNN model achieved an accuracy of 99.5%, while the SVM model achieved an accuracy of 100.0%. The SVM model was able to achieve this high accuracy by using the Histogram of Gradients (HOG) feature extraction method to capture the shape and orientation of the edges in the images. The CNN model, on the other hand, was able to achieve high accuracy by directly ingesting the raw pixel values of the images and automatically learning relevant features through the convolution layers. Both models were able to effectively classify the images into their respective classes, but the SVM model may be better suited for tasks that require dimensionality reduction and computational efficiency, while the CNN model may be better suited for tasks that require robustness to noise. Future work could involve further tuning the hyperparameters of the models, exploring different feature extraction methods, and evaluating the models on larger datasets to further improve their performance. Overall, the results of this project demonstrate the effectiveness of machine learning models for classifying the ASL fingerspelling alphabet and provide valuable insights into the strengths and limitations of different machine learning models for this task.

References

[1] A. Nagaraj, "Asl alphabet," 2018. [Online]. Available: https://www.kaggle.com/dsv/29550