American Sign Language Letter Classification through Support Vector Machines

Machine Earning Group
EEL4930 4/19/2020

***Abstract*— This study aims to create a machine learning model to classify American Sign Language letters A through I to achieve the highest accuracy possible. To create the model, Scikit-learn libraries were used and the primary dataset used for training was 1844 images collected by Dr. Silva's machine learning class. The support vector machine displayed the highest performance and benefited from training on offline images and identical rotated images. Histogram of Oriented Gradients and Sobel edge detection were instrumental in feature generation. The resultant model achieved a very high accuracy near 100 percent.**

***Index Terms*— American Sign Language, Histogram of Oriented Gradients, Supervised Learning, Support Vector Machine**

I. Introduction

THE topic of this study includes utilizing machine learning classifiers and strategies to classify images of American Sign Language (ASL) letters A through I, achieving the highest test accuracy possible. Reviewing previously conducted research papers show support vector machines (SVM), multi-layer perceptrons (MLP), and convolutional neural networks (CNN) as promising approaches towards the solution. An accuracy of 90.3 percent was achieved with a CNN by using 50,000 ASL images for training the model [1]. Moreover, an SVM outperformed an MLP model's accuracy by 9.44 percent overall while showing more rapid training [2]. To discriminatively classify, an SVM works by separating classes with a manifold.

Because the study was limited to 1844 ASL training images similar to the testing image set, a CNN would require further training data collection for optimal performance. With limited time to train a model, SVM offered a robust solution, not requiring extensive data collection. It should also provide more rapid testing than a CNN which may require many epochs of training to achieve a high accuracy. When using a SVM for ASL image classification, image thresholding to extract significant features showed high value to reduce noise and parts without value [3].

II. Implementation

To begin classifying ASL characters, some initial data is required to train and test the classification model. For this classifier, the data used consisted of 100x100 pixel images, varying from characters A to I in ASL, and various person's hand signing each character five times for a total of 1844 images. Associated with each image was the corresponding ASL character label.

Once data was collected, new images were created by rotating the images of the original data by -5 to 5 degrees to augment the size of the dataset and improve model performance. Then, the images were converted from RGB (red, green, blue) values into grayscale so that color wouldn't contribute to the classifier's results. Next, the Sobel edge detection algorithm was used to emphasize the edges of each hand. A Histogram of Oriented Gradients (HOG) was applied to the Sobel images to generate features that would be used to distinguish between the different characters (Fig. 1). Along with generating features, the HOG was also used to standardize the images so that the images shared a common range of pixel intensity because using typical standardization techniques caused the outputted images to overlap with others, leading to a lower accuracy, discussed below.

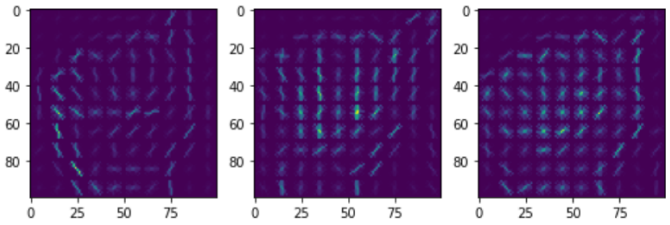

Fig. 1. Histogram of Oriented Gradients for the ASL letter A. The blank dark surroundings show little gradient change in the background, putting more focus on the hand's features. The grids are 100 by 100 like the image's pixel size.

Next in the pipeline was truncated singular value decomposition (SVD), which helps reduce the dimensionality of the feature space, avoiding the curse of dimensionality and reducing the file size for accessibility purposes. The next execution of the pipeline would be the classification model. The classifier SVM was used because it performed best among different classification models. Specifically, the SVM parameters used were C, kernel, class_weight, gamma, and break_ties.

The parameter C is used to control the cost of misclassification on the training data, meaning if the value is small, the classifier will allow some data to be misclassified. However, a large C value tries to force the algorithm to learn

the data correctly and possibly overfit. The kernel parameter is used for pattern analysis and data transformation. The class_weight parameter gives all classes a weighted value to emphasize how much the classifier should learn each class. Gamma determines how much influence a data point has by its distance such that it can affect the decision boundary. The purpose of the break_ties parameter is to classify an input based on the associated confidence value if it is stuck between classes. After fine tuning the parameters, the parameter values that performed the best were C = 20, kernel = RBF, class_weight = balanced, gamma = "scale", and break_ties = True.

Since the model can only classify ASL letters A to I, additional changes were made to classify other ASL letters or other objects to an unknown class with label "-1". To do this, the model was augmented with an ability to also output the probability that each output's label is the true label. Since SVM is not equipped with this functionality as a CNN with a softmax output layer is, these probabilities are generated through cross validation. Thus, this approach may yield a different result than the SVM's typical prediction. Incorporating this functionality, the model first determines if the probability for any output is greater than 40 percent. If not, the input is classified as the unknown class; otherwise, it outputs the predicted label.

III. Experiments

Main Model Selection

Before selecting SVM as the main classification model, other models were evaluated and compared. These models included K-Nearest Neighbors (KNN), Random Forest, LDA, MLP, and decision trees. Parameters were tuned with a gridsearch for each classifier, indicated as follows: for KNN: number of neighbors and distance metric; for random forest: number of trees, minimum number of samples required to be a leaf node, and maximum tree depth; for LDA: solver type; for MLP: activation function, solver type, batch size, initial learning rate, hidden layer sizes, number of neurons in each hidden layer, and momentum; for decision tree: minimum number of samples required to be a leaf node and maximum tree depth. The SVM showed the highest training and testing accuracy through this initial comparison (Table I) and was thus used as our primary model.

TABLE I
Initial Model Comparisons

Model Type	Training Accuracy %	Test Accuracy %
SVM	100	83
MLP	97.9	82.4
KNN	100	81
Random Forest	99	57
LDA	67	53
Decision Tree	100	44

Improving SVM

With a more expansive tweaking of SVM parameters through a gridsearch, a test accuracy of 89 percent was realized with parameters discussed in the implementation section. To improve model performance, we tried additional feature generation with the Histogram of Oriented Gradients (HOG), improving test accuracy up to 94 percent. HOG converts the image into its corresponding gradients, emphasizing the lining of the fingers and palm which are the critical features in forming an ASL character.

Adding Rotations

Data collection is a critical step in machine learning applications as additional data improves model performance significantly. Hence, data augmentation by a variety of methods such as rotating images, squishing them, stretching them, and resizing them contribute greatly to model performance. We first implemented rotation with the intention to do additional data augmentation if necessary. Each image was rotated by -5 to 5 degrees, expanding the dataset by an order of magnitude. After training the network on this new dataset, which took significantly longer, test accuracy increased to 100 percent. The boost in performance indicated that additional data augmentation would not be necessary. While this may be an indication of overfitting, the test data that will be fed into the model is very similar to the data it was trained on, as pictures were taken in the same context with little variation in hand movements. Hence, this accuracy was reasonable to attain. Nevertheless, we underwent additional experimentation to ensure these were not errors, as explained next.

Kaggle Data Experimentation

To ensure the 100 percent test accuracy was valid, the model was trained on three different datasets and compared against each of the others in test. The first two datasets included the original dataset and the ten times larger rotated dataset. The third came from the Kaggle ASL classification competition. The first model was trained on the rotated dataset and performed with a 100 percent accuracy on the original dataset but with a 15 percent accuracy on the Kaggle dataset. The second model was trained on a split version of the original dataset and performed 98 percent on the corresponding test set, while performing 15 percent on the Kaggle dataset. The last model, trained on a split version of the Kaggle dataset, performed only 14% on the original data but with 99 percent accuracy on the corresponding test set of the Kaggle dataset. These results indicate that the model was in fact performing properly and would classify datasets with an accuracy equal to that as expected. A model trained on the original dataset and tested on Kaggle with poor performance, or vice versa, is expected as the Kaggle dataset includes ASL characters that are signed differently than those in the original dataset, such as letters 'c' and 'e'. Moreover, the background of the Kaggle images differed from our own. Additional datasets were created by taking images of ASL characters against different backdrops such as a black curtain and a messy room with accuracies above 80 and 90 percent. Thus, the model performed well on the original dataset where the test set is very similar to the training data, but did not generalize as well to other images. Later

experiments described below rectify these issues, but first a note on normalization and the size of the model.

Normalization

The final model does not include normalization explicitly. Originally, the model was normalized with standard scaling; however, the fact that the mean of the dataset is subtracted from each image caused shadowed versions of other images to be overlaid on top of the original image, deteriorating performance. When the scaling was removed, accuracy improved. The normalization could have been replaced by min/max scaling, but HOG includes a normalization component in its transformation, making it redundant. While normalization is often an essential feature of a ML pipeline, the metric for success is accuracy; since accuracy was greater without normalization, it was removed.

Dimensionality Reduction

The model trained on the ten times larger rotated dataset caused the model to be as large as 0.5 GB. For accessibility/transportability issues, the model size was reduced significantly. To accomplish this, singular value decomposition (SVD) was utilized. This feature extraction technique provided better results than PCA. Furthermore, several dimensionality reduction techniques were utilized including incremental PCA, fast ICA, sparse PCA, and truncated SVD. This last feature extraction method had the greatest effect on performance while reducing file size significantly and was thus used.

Sobel Edge Detection

Performance on the easy test set which includes images very much like those in training was now likely to be high. To improve performance on a dataset that included additional ASL characters outside the range of A-I and additional objects that were not ASL characters would require additional feature generation. Thus, we added the Sobel edge detection algorithm. This applies a kernel to the image that highlights where in the images there are edges. This output was then fed into HOG as discussed previously. The Sobel kernel greatly improved performance on harder test sets. Additional pictures were taken, one set against a black curtain and another against a messy room where there were many objects. With the Sobel edge detector, accuracy on the black curtain images was 100 percent. Hence, the model now generalizes well to images that are outside the training data and is more likely to do better on new images in a harder test set. Similarly, performance on the ASL characters against a background of many objects performed better as well with the introduction of Sobel, at 88 percent. This performance isn't quite as high as it is for other datasets, but this is because the training data does not include backgrounds with a wide array of objects. Thus, it did not have the ability to learn to discern the ASL edges from that of the others. Nevertheless, the ability of Sobel to detect edges intuitively allows the model to find an ASL character in the image, despite there being other objects.

Moreover, the Sobel edge detector was compared with the Roberts edge detector, a similar kernel. The results with this edge detector were identical, so either choice of kernel would have performed equally. However, the Sobel detector introduced less noise into the resultant image, motivating the decision to use Sobel.

Furthermore, we experimented with using just Sobel versus using Sobel and HOG. Since on the messy background, the Sobel edge detector could cleanly show the outlines of each of the objects in the image, while HOG would instead appear to be an incomprehensible array of edges, we believed Sobel alone would have performed better. However, when removing HOG from the pipeline, accuracy deteriorated. Hence, we included both feature generations in the pipeline. The reason for the drop-off in performance is unknown, but one reason may be due to the normalization that HOG provided. Trying Sobel alone with a normalization technique was not tried, and may be a point of new research.

Predicting Unknown Objects

While the model could classify ASL characters between A and I very well, even among different backgrounds, an additional class named unknown had to be created to classify characters outside the range or to classify altogether different objects. The natural way to do this is to examine the probability that an output is likely to be the true label; if below a certain threshold, then the input will be classified as an unknown object. While this method works particularly well with ANNs with a softmax output layer, SVMs do not have this probabilistic determination as it simply finds hyperplanes to separate classes into regions. Thus, a cross-validation method implemented in scikit-learn was used to determine probabilities. This method compares the given input to several outputs with its predict function and generates the proportion of the time it is classified correctly, which is converted into probabilities. It can then output the probability that the output matches the true label. In our model, if no output probability exceeds a certain threshold, it is classified as unknown; otherwise, the predicted label is assigned. Experimentation with different inputs was done to determine this threshold value, which was decided to be 40 percent. At this value, ASL characters of A through I are never classified as the unknown class. At the same time, non-ASL objects are classified as unknown, such as water bottles and other small, simple objects. Given an ASL character outside the A-I range will result in a probability greater than 40 percent, and thus will be misclassified. This is expected as the training data does not include ASL characters outside the range. However, raising the 40 percent threshold would misclassify ASL characters in the A through I range against a complex background. Additional research with a larger dataset of characters and unknown objects could shed light on the threshold value necessitated.

IV. Conclusion

In conclusion, the trained model and test model accuracy were both 100 percent. This is due to the methods of data collection, where little variation between images existed as

pictures were taken in quick succession. Despite this, the model will likely perform very well on the withheld test set as it closely resembles the training data. Nevertheless, the model does indeed generalize well. On new images taken in front of a black curtain, accuracy is 100 percent, while in front of a messy background it is 88 percent, a reasonable accuracy. The key components leading to the high accuracy of the model included the rotation of the images, augmenting the training set by ten times. HOG and the Sobel edge detection method were also critical in classifying objects in a more varied test set. Nevertheless, to improve the generalization of the model, additional data with different kinds of backgrounds could be included, serving as a new starting point for additional research.

REFERENCES

[1] Daroya, Rangel & Peralta, Daryl & Naval, Prospero. (2018). Alphabet Sign Language Image Classification Using Deep Learning. 0646-0650. 10.1109/TENCON.2018.8650241. https://www.researchgate.net/publication/331854265_Alphabet_Sign_Language_Image_Classification_Using_Deep_Learning

[2] M. Ugale, S. Bende, M. Haldankar, S. Hedge, "ASL Fingerspelling Interpretation Using SVM and MLP," International Research Journal of Engineering and Technology, vol. 5, no. 3, Mar. 2018. Available: https://www.irjet.net/archives/V5/i3/IRJET-V5I3410.pdf

[3] D. Mali, N. Limkar, S. Mali, "Indian Sign Language Recognition using SVM Classifier," International Conference on Communication and Information Processing, 2019. https://poseidon01.ssrn.com/delivery.php?ID=282093115122109089124118098015088069019088031054017090087071080000084099066003070103023026019052022005116112112081110065114109011066028011020097082116117092116080088086013051126064098000004089124081119003080030072073089084094002024113074020096065095125&EXT=pdf

[4] Tecperson, "Sign Language MNIST," kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/datamunge/sign-language-mnist.