Project Final Report

James Galante and Thomas Killian

Note: Real-time classification (ML_Project_Mediapipe.ipynb) only works with Jupyter Notebooks due to compatibility with the webcam; the other notebook(FinalCopy.ipynb) was created and should be run in CoLabs

Initial Problem Statement

Initially, we hoped to use two Kaggle American Sign Language datasets to train an image classifier using various machine learning techniques. The image classifier would take in an image or a csv file of pixel values and return the associated alphabet letter. We did not include the classification of J and Z because they require movements. Initially, as stated in our proposed milestone, we hoped to classify the dataset using K Nearest Neighbors, Support Vector Machines, Naive Bayes, and possibly Neural Networks. This would hopefully lead to a real-time classification using mediapipe hands. At this point there was discussion on whether or not we would use real time classification to classify J and Z. Furthermore, if we could incorporate movements into our classification, we might be able to classify words using a dataset we found.

All of this would be in an effort to make a possible educational platform that would determine the accuracy of someone trying to learn sign language. There was even a discussion about making an ASL wordle, which would've been the same as the original wordle, however, it would use our real-time classification program.

Updated Problem Statement

Our initial problem statement tended to stay on track with the eventual hope of real-time classification, however, we segued into real-time classification after heavily looking into Convolutional Neural Networks and its associated libraries. After doing research, we found that using a CNN to classify American Sign Language would give the most accurate classification possible. This would allow us to continue with our initial problem statement and goals of creating the best possible real-time classifier for ASL. The goal of this real-time classification is still to provide an accurate platform to practice and learn the ASL alphabet. All other factors of the initial problem statement are included in the updated problem statement.

Data Description and Information

At first we were deciding between two different Kaggle datasets to run our classifications. After running some initial tests to complete our milestone, we decided on one

Kaggle dataset over the other, which is linked below. However, this posed some problems when we shifted our focus towards real time classification. In order to use media-pipe hands and train a classifier based on the landmarks that media-pipe hands assigns, we needed to fit media-pipe hands to all of our images. We eventually found another dataset that had already processed images to do this, so we used that dataset to train our real-time classifier.

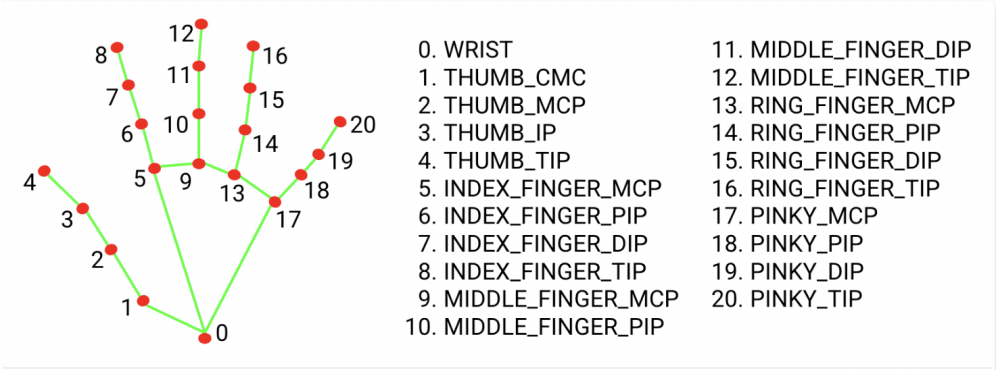
- The Kaggle Dataset we initially used for SVM, KNN, CNN, and Naive Bayes is a CSV file of ~34,000 samples. Each sample has 784 features, which are correlated to each pixel in the 28*28 pixel images. We used this dataset by adding it to our google drive folder, and importing it by mounting our google drive folder to google collab. The location of the appropriate files in our program must then be changed to run the program in a separate local computer.
    - Sign Language MNIST Kaggle Dataset
- When we moved to MediaPipe Hands the data either had to be transformed or a new dataset had to be found or created. Our project ended up with the latter as we discovered a new dataset. The original dataset was provided by Kaggle and this new dataset was created and stored on GitHub. This dataset converted a set of 1,000 images per sign into 21 landmark arrays. These 21 landmarks represent the x and y coordinates returned by mediapipe hands when tracking an image. This dataset is then used to both train and test a KNN and SVM to then perform real-time classification in MediaPipe Hands. In order for MediaPipe Hands to accurately classify your sign you must use your left hand as that is what the dataset contained. This new dataset has some signs that differ slightly from our original so included below is both the dataset along with an example of each letter's sign that was used to develop the 21 hand landmarks.
    - MediaPipe Hands Dataset
    - MediaPipe Hands Images
    - Original Kaggle Dataset (Pre-Transformation into Landmarks)

Systems and Programs

For our project we used a variety of programs and techniques. For the initial machine learning techniques (SVM, KNN, Naive Bayes, and PCA) we used Sklearn, Pandas, and Numpy, and Matplotlib. These four libraries provided us with all the functionality we needed to manipulate the dataset, and train and test our models.

After doing research on image classification using machine learning methods, we found that Convolutional Neural Networks were invaluable. To use a CNN, we had to first learn how they worked and secondly how to implement them using standard libraries. We started off with pytorch, but after running into difficulties, we decided to use tensorflow in combination with Keras. These libraries were relatively easy to use and the documentation online was plentiful.

Finally, we decided to use mediapipe with our classification techniques for real-time classification. MediaPipe hands employs around 21 feature landmarks on a hand by using machine learning methods. These landmarks are 3D and are the crux behind our real-time image classification. An image of each landmark and its position can be seen in the image below.



Experiments

Our experiments with the ASL datasets we used were initially limited to classical machine learning techniques like K-Nearest Neighbors, Support Vectors Machines, Naive Bayes, and Principal Component Analysis. We used sklearn to implement all of these algorithms and used matplotlib to visualize the PCA results. We found that PCA turned out to be not that useful, and this was an important lesson in our approach to machine learning, which will be discussed in the conclusion section. For each of these models, each sample had 784 features, which lined up with each pixel in the image.

Past traditional machine learning techniques, we looked further to better classify our models based on image alone (at this point we were not using MediaPipe Hands to indicate landmarks). All of our research led to Convolutional Neural Networks. First we had to learn about CNN's and we used a bunch of online resources to do so. Youtube was of great help in this regard. Furthermore, we had to learn how to use the appropriate libraries. Initially, pytorch was of great interest because we had experience with pytorch and linear models, however, tensorflow and keras ended up being our preferred choice.

Our initial CNN was run with >10 Convolutional layers and a resulting 4x4 image that was run through a dense neural network. This resulted in a lower accuracy than our KNN and SVM models. After more research, we found that decreasing the amount of layers to around 3 or 4 convolutional layers worked a lot better. At this point the accuracy of our CNN was around 93%. Interestingly, each time we trained the model without changing the CNN architecture, we arrived at a different accuracy percentage. The next problem we ran into was overfitting.

After even more research we found that we could bring our accuracy up to a range of 96%-96%. We were able to do this through a few different changes. After sticking to only a few convolutional layers, we learned about Max Pooling, Average Pooling, and Dropout. These were all very helpful techniques. Also using a specific activation function called ReLu helped decrease computational power and reduce the probability of vanishing gradients, which is what happens in neural networks when an update via the loss function doesn't propagate to the beginning layers very well. Learning the math behind all of these techniques was also very interesting. Finally, we applied image transformations to our training dataset so that the model would hopefully generalize better.

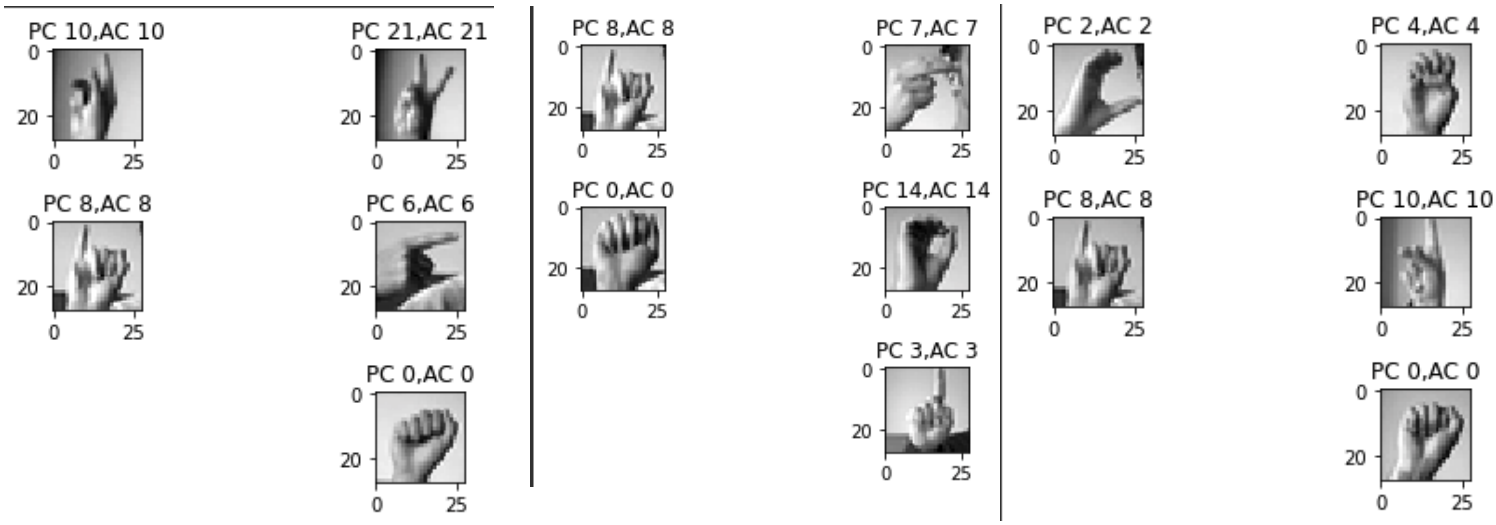
While the classification of the ASL dataset using a CNN proved to be useful in its general context, the CNN turned out to be less useful when it came to real-time classification. For real-time classification it is necessary that the classifier be able to predict your result in that instant and the CNN would have required the exportation of the images instead of true real-time classification. Because of this, our real-time classification pivoted back to the KNN and SVM models from before, but with a different dataset. This new dataset correlates back to the resulting list of landmarks returned by MediaPipe Hands and allows for much more accurate classification by the KNN and SVM classifiers. One minor flaw with our new dataset stems from the fact that the way the hand landmarks were obtained one must use their left hand to perform the signs and get accurate predictions. Furthermore, the distance from the camera matters and there is certainly a sweet spot approximately one and half feet away from the camera. Overall, the dataset used for MediaPipe hands took in a new input of 21 unique x and y landmark pairings (totaling 42 features) and had approximately 800 data objects per label (A-Y excluding J). The output of this was still the same with the prediction of the label or letter that was being signed. In total, the dataset consists of 1000 data objects per label, but the training_size was 0.80 meaning that around 80 percent of each label's data objects were used for testing. Ultimately, this new dataset drastically improved the accuracy of KNN and SVM and provided for a wonderful model that does a fairly accurate job when it comes to real-time classification.

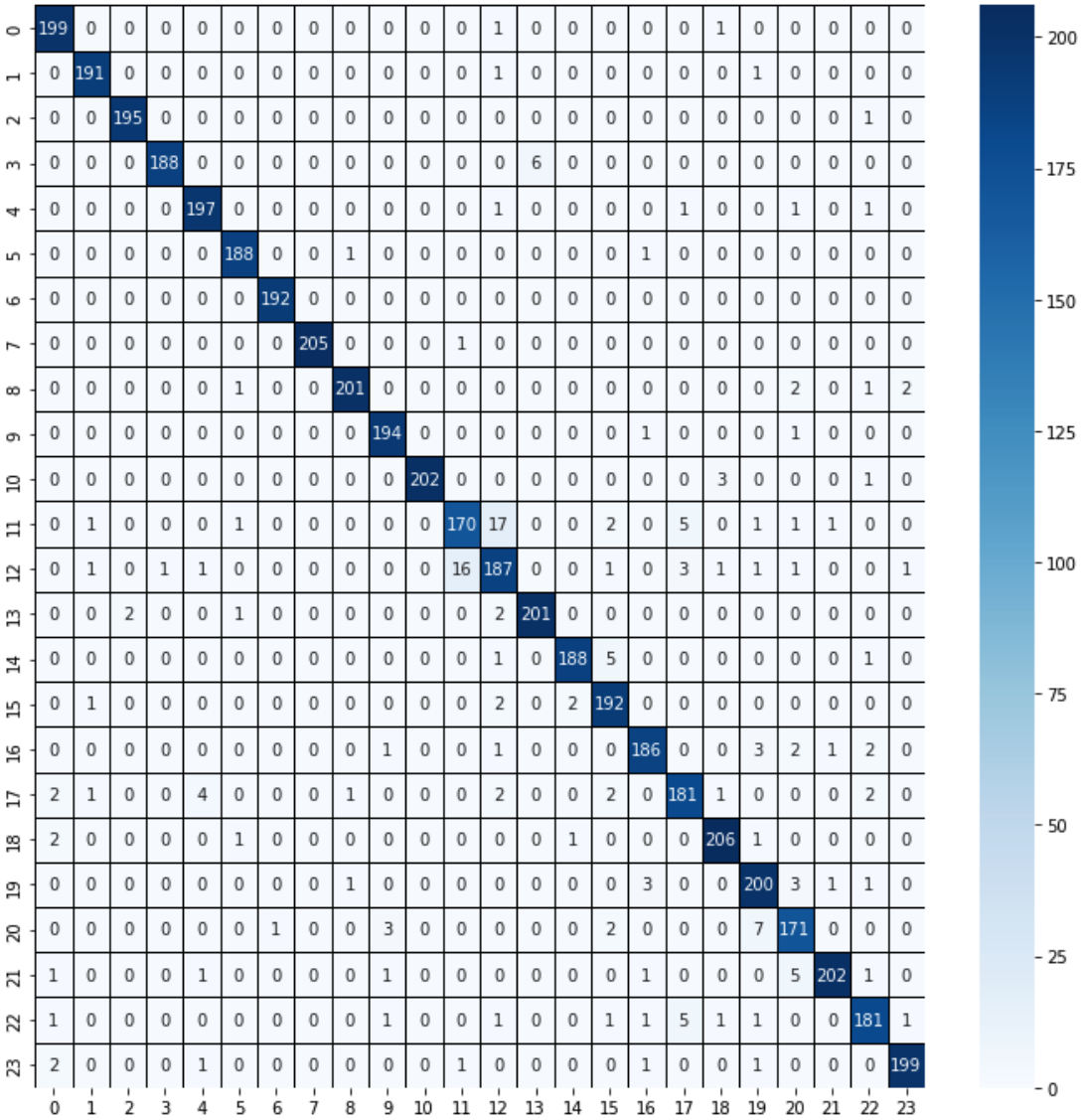
Our criteria for performance was at first purely relative. In short, if one model did better than the next, then this was a success. Because we were already at a high accuracy from the start, increasing accuracy turned out to be rather difficult with the CNN. Once we were able to get accurate models with just KNN and SVM models, this was an even bigger success, and only correlated to how we prepared the data. However it's important to note that MediaPipe Hands already uses machine learning techniques to find landmarks.

Experimental Results

- Accuracy of each SVM, KNN, Naive, CNN on the Kaggle Dataset
    - SVM: 85%
    - KNN: 81%
    - Naive Bayes: 39%
    - CNN: 93% -> 96-99%
    - As can be seen in the numbers above, the CNN seemed to be the clear winner in terms of accuracy, so we spent a lot of time trying to refine the CNN. SVM was the second best followed by KNN and then Naive Bayes.
- SVM and KNN with MediaPipe Hands Dataset
    - SVM: 94-96%
    - KNN: 95-97%
        - This discrepancy in accuracies is due to the shuffling when calling train_test_split()
    - Interestingly with the MediaPipe Hands Dataset, the SVM and KNN classifiers performed at the level of the CNN. This was really important for our real-time classification, and mainly stems from the use of a new dataset oriented around the hand's landmarks.

- Below are some images of the CNN Classifier with the predicted class (PC) and the actual class (AC). As can be seen with all the data points, the actual class and the predicted class are the same.





- Shown above is the Confusion Matrix for the K-Nearest Neighbors Classifier that is ran within the notebook containing MediaPipe Hands. As you can see, the predictions are incredibly accurate, but still have flaws that are easily visualized here.

- As you can see, the real-time classification is very good at classifying letters that might look very similar (A, E, S). Overall the model is very good at classification if the hand is at the correct distance from the screen and the left hand is used. The bottom right example shows that some letters (F) are still misclassified.

Conclusions

We learned a lot of techniques and information on classification throughout the length of our project. This involved a lot of information about overfitting and how to combat that by generalizing our model. This was discussed thoroughly in the experiments section. Pooling, Dropout, and Batch Normalization were some of the techniques we used. We also got used to a lot of libraries that we wouldn't have known about before. These included tensorflow, pytorch, and keras. Trying to figure out how to use all of these was a struggle, but being able to put together complex neural networks was really easy once we got the basics down. IN terms of classification, we hadn't thought about taking the average or using two different classifications for real-time classification.

One important thing we learned when we tried to use principal component analysis and through the initial use of KNN's and SVM's was that just throwing machine learning techniques at data doesn't work. We tried to visualize our data with PCA, and this didn't prove to be that helpful. Interestingly, the data formatted through MediaPipe hands performed much better in

KNN and SVM than just using the pixel values of the images. In this way, how the data is prepared is extremely important in classification techniques.

While CNN's were interesting to look at and learn about, I think for the purposes of our project we could've just focused on increasing the accuracy of our KNN and SVM with the data prepared by MediaPipe Hands. We ended up only using those classifying techniques in the end, and they were nearing the classification accuracy of the CNN we trained. It would also be helpful to read more into different techniques before just jumping into them. We found that we could easily implement different models, but at some points we were a bit confused on what was happening (specifically with CNN's). We also learned that less is better with CNN's, which was a very valuable experience.

Some further directions with this project could include incorporating J and Z into the real time classification. We think this would mean using a time-series model. We looked into time-series models, but had trouble finding a good fit, and incorporating it into our program in the time span of the project. Also, if we were able to begin using words, with time-series analyses, we would learn a lot about sign language along the way, and be able to further the program for educational purposes.