**EECE 5644: INTRO TO MACHINE LEARNING AND PATTERN RECOGNITION – V30**

**PROJECT FINAL REPORT**

**ANUSH SRIRAM RAMESH,** [*ramesh.anu@northeastern.edu*](mailto:ramesh.anu@northeastern.edu)

**KYUNGSUN LEE,** [*lee.kyu@northeastern.edu*](mailto:lee.kyu@northeastern.edu)

**SAGE TRUDEAU,** [*trudeau.s@northeastern.edu*](mailto:trudeau.s@northeastern.edu)

**UKHYEON SHIN,** [*shin.uk@northeastern.edu*](mailto:shin.uk@northeastern.edu)

**MOTIVATION**

As the world moves to more and more connected interactions with voice assistants now in our homes, cars, and phones, the hearing-impaired community is left without a reliable means of interaction. This creates an additional boundary to overcome when looking to benefit from some of the great features of the rapidly evolving virtual personal assistant space. In the last decade, there has been incredible progress in vision-based systems hand recognition capabilities. Our team recognized this as the key enabler for a ASL recognition capability. The community working in this space has provided some fantastic resources such as the [Kaggle ASL dataset](https://www.kaggle.com/datasets/grassknoted/asl-alphabet). Our team hopes to build upon this work and utilize the techniques learned in class to highlight some of the fundamental problems in this space, as well as what we see as best practices given the nature of the data. By utilizing our data analysis skills, our team hopes to highlight the most promising ways forward and contribute a plausible methodology to achieve the goal of real time human-machine conversation – in silence!

A collage of a person's hand

Description automatically generated with low confidence

**Fig 1: American Sign Language (ASL) Images**

If this work is successful, it could open the door to new interaction possibilities for unimpaired people as well. The work to determine ASL could be expanded to recognize additional hand gestures beyond the English alphabet and provide a method of human-device interaction which does not need auditory response. A light could turn on simply with a point of a finger, or an action could be performed without disturbing a baby sleeping in the room.

**GOAL**

Our goal with this project is to build a classifier which learns to model each of 29 classes (26 English alphabet, blank, space, and null classes) as uniquely identifiable to a machine. By identifying and mathematically describing what aspects of a picture are unique to that class we hope to create an algorithm capable of transcribing imagery of ASL. Our team has further divided this into two sub-goals in order to systematically approach this problem.

The initial milestone is identifying the classification model that best suits this dataset and train it. It will be important to take a wide approach and try many solutions because we expect many of our approaches to fail. As we evaluate model candidates, we must maximize accuracy while carefully avoiding common pitfalls of machine learning models such as overfitting or unintentionally modeling non generalizable biases that could be present in our training data.

A more ambitious goal is to create a model which is generalizable beyond our initial training data. If this is successful, it could enable this classifier to broaden its inputs and enable live online classification in the stream by dynamically locating the hand and labeling it with the class predicted.

**APPROACH**

Timeline

Description automatically generated

**Fig 2: Gantt Chart for the project.**

The approach we inherit to solve this problem is as follows:

1. Fetch the dataset and perform EDA on the dataset.
2. Test supervised learning classifiers by tuning the parameters to suit the dataset. The methods we used for this dataset are:
   1. Naive Bayes classifier
   2. Logistic Regression
   3. Decision Tree/Decision Forest
   4. Convolutional Neural Networks
3. Test unsupervised learning classifiers on dataset to analyze their performance. Cross-validation and regularization will be leveraged to help minimize overfitting. The models used are:
   1. K Means Clustering
   2. Convolutional Neural Network
4. Preprocess the image dataset to reduce the high dimensional image data to lower dimensional numerical hand descriptions, which should help the model generalize better to unknown data.
5. Propose the results from each classifier algorithm and choose the best one to pursue for the final solution and attempt to extend that classifier to unknown datasets.

**DATASET**

The initial training dataset consists of 87000 images in the training folder and 1 image for each class. Each class contains 3000 sample images with each image being of the shape (200,200,3), which means each image is 200 pixels wide and 200 pixels tall with 3 values in each pixel, constituting a RGB image. The dataset does a good job of providing samples of hand with different orientations, sizes, and lighting conditions. A test dataset is also provided with this training dataset of 1 image from each class. Due to a lack samples in the test dataset, the team opted to instead implement an 80:20 split in the training data to hide more data from the model while training to use for validation and testing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | low light far away | low light closeup | normal lighting | high contrast |
| raw | A picture containing person  Description automatically generated |  |  | A close-up of a hand  Description automatically generated with medium confidence |

**Fig 3: Sample images from the training dataset showing the “A” class.**

**PRELIMINARY RESULTS**

In the table below, the accuracy of each approach is reported. The image data was modeled as is without any preprocessing to generate our baseline expected performance.

|  |  |
| --- | --- |
| **Technique** | **Accuracy (%)** |
| **Naïve Bayesian with Bootstrap method** | 46.24 |
| **Naïve Bayesian with 5-fold Cross-validation** | 45.33 |
| **Multi-nominal NB** | 53.9 |
| **Complement NB** | 38.09 |
| **Logistic regression** | 75.27 |
| **K-means clustering** | 69.65 |
| **Decision Tree** | 90.4 |
| **Decision Tree with 10-fold Cross validation** | 98.0 |

**Table 1: Accuracy of classification techniques with original data**

For our initial comparison of models, accuracy was chosen as our primary metric. As expected, due to the large variations in lighting and range, Naïve Bayes approaches performed with low accuracy. Log-reg, and K-means performed decently well, but there is suspicion that they may be benefitting from overfitting and may not be transferable to a real-world situation. For the Decision Tree model, we initially let the tree grow until it is impossible to split. Using this model, we predict the label on the untrained data.  This can be seen in Figure 4 below. The tree is too complex with more than 5000 nodes and max depth of 36. The tree achieved an accuracy of 90.4%, but this is likely too good to be true.

Diagram, schematic

Description automatically generated

**Fig 4: Result of decision tree without pruning**

**CNN Results:**

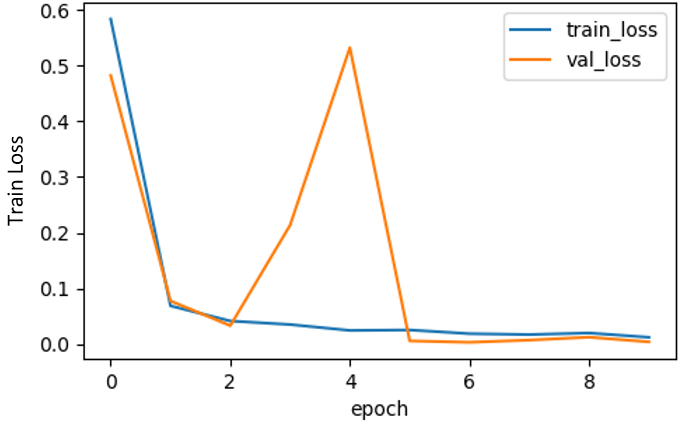
The initial unsupervised results were a promising start, but we had read about work utilizing CNNs to accomplish hand recognition and so we wanted to implement an unsupervised model as well to compare.

|  |  |
| --- | --- |
| **Dataset** | **Accuracy (%)** |
| **Original dataset** | **99.8** |
| **Own pics before preprocessing** | **18.4** |
| **Own pics after preprocessing** | **Revealed in Table 4!** |

**Table 2: Accuracy of CNN with different dataset(original dataset vs. Own pics)**

For our CNN we started with a training example from scikit-learn documentation. Our first 3 layers are Conv2D layers paired with max pooling. These are convolution layers that deal with our input images, which are seen as 2-dimensional matrices. 64 in the first layer, 128 in the second layer, and 256 in the third layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. We experimentally determined that 64, 128 and 256 work well, so we will stick with this for the test. We then utilize normalization, flattening and dropout, before finishing with two dense layers. 10-fold cross-validation is also used to help avoid overfitting.

Chart, line chart

Description automatically generated

**Fig 5: (left) Accuracy of CNN with original dataset. (Right) Loss of CNN with original dataset**

The activation function we used for our first 3 layers is the ReLU, which has an advantage to avoid vanishing gradients problem and provides a much faster run time. This activation function has been proven to work well in neural networks such as the CIFAR image processing challenges where it became popular in its implementation of the AlexNet CNN. In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers. Our final 2 layers are dense layers. Dense is the layer type we used in for our output layer. Dense is a standard layer type that is used in many cases for CNN. We have 1024 nodes in our first dense layer with sigmoid activation function. Then the last output layer has 29 nodes, one for each possible outcome (A–Z plus nothing, delete, and space classes). In the output layer we used SoftMax. SoftMax makes the output sum up to 1 so the output can be interpreted as probabilities. The model makes its prediction based on which option has the highest probability.

The test results were interesting. Using the given the Kaggle ASL dataset, we were able to have remarkably high accuracy, but it caused concerns about whether this model would ultimately suit our real-world imagery goals. Hence, we conducted a second trial that tested against our own self taken images and saw as suspected a huge loss in accuracy to 18.4%. This prompted us to prioritize our efforts into preprocessing the data to achieve a more generalizable model. This led to some exciting results which we will share along with the preprocessing breakthrough in the following sections.

**PICKING THE BEST APPRAOCH**

Figure 6 below shows processing time versus accuracy of all of approaches using preprocessed data. As the training time between runs varies slightly, we used mean value of time and accuracy, dividing the value into the number of epoch or iteration. Top-left side is the best result which has shortest time and highest accuracy among all approaches. However, in real experiment, we can see trade off relation between time and accuracy. The best accuracy was given by CNN, on the other hand, best processing time was given by Gaussian Naïve Bayesian. In the case of the decision tree, pruning decreased the model’s performance which further supported the hypothesis that it was just overfit rather than being a good model.

Table

Description automatically generated with low confidence

**Fig 6: Accuracy vs. Time trade-space of each approach with preprocessed data**

Overall, we were quite happy with our initial results. The decision was made to focus our efforts on our CNN and have log-reg as a backup as they were providing the best results and it was decided that all the preprocessing times were acceptably fast. With our algorithm approach decided, our teams focus shifted from model accuracy to instead maximizing the model’s ability to generalize to all the variance in real world imaging conditions. In order to pursue our goal, the team felt that a preprocessing step was our best approach to improving the capabilities of our models.

**PREPROCESSING**

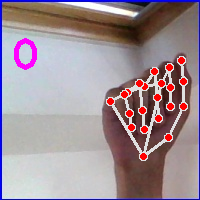
With our algorithm design decisions made, we focused on what domain knowledge we could leverage to improve our model’s success. We had recognized a few issues in our training data, and we wanted to expand our approach to generalize beyond these. Prominently, these issues were varying lighting conditions, unknown scaling, and orientation of the input at test time, and lack of ethnic diversity in the training data. From our webcam experiment, it was seen that the CNN results suffered heavily when the input image came from an unknown dataset. To attempt to remedy this, an initial preprocessing step was implemented to perform shape detection and scale the input to just the area of the image associated with the hand. This assisted in rectifying the first two problems, but due to its thresholding approach it did not help with diversity of skin tones.

Text

Description automatically generated

This was a promising start, but ultimately an unacceptable answer for our final implementation. With this knowledge in mind, we broadened our approach from just shape detection to specifically hand detection to leverage other gesture recognition work in the community and bring its applications to ASL.

We were able to utilize the media pipe library to accomplish this goal. The media pipe library utilizes a series of image processing techniques to determine 20 “landmarks” associated with each hand. These landmarks record the X and Y position of the landmark in the image, as well as an approximate depth Z from image gradient. This changes the domain of our classifier input from a 2d image of 200 by 200 pixels to a single column vector of length 63 where each x, y and z of each of the 20 landmarks is input to the matrix. A visualization of the two approaches can be seen in Figure 7 below.

Icon

Description automatically generated**Fig 7: Original thresholding approach verses feature extraction example of Media pipe**

This was very effective but still left us with an issue of relative scale. We handle this by transforming our datapoints from the original image coordinate frame to a scaled relative coordinate frame with reference to the wrist. This further removes the data’s specific dependance to our training dataset, while preserving the information of relative finger positions which is paramount for ASL.

**RESULTS AFTER PRE-PROCESSING**

We tested and compared each techniques’ accuracy between BEFORE preprocessing data and AFTER preprocessing data to figure out the merit of the preprocessing.

테이블이(가) 표시된 사진

자동 생성된 설명

**Table 3: Accuracy comparison of each approach with before preprocessing data and after preprocessing data**

A picture containing wall, indoor, person, game

Description automatically generatedA picture containing wall, indoor, person, game

Description automatically generatedA hand with a thumb up

Description automatically generated with low confidenceA picture containing text, wall, indoor, flat

Description automatically generatedA collage of a person's face

Description automatically generated with medium confidenceIn the case of CNN technique, preprocessed data made the accuracy slightly worse, but now the model’s training input was much more generic and so we hoped it would be more flexible. In other cases, the preprocessed data was beneficial for naïve Bayesian and log-reg. For the decision tree case, an accuracy of 89% was achieved. However, the tree remained too complex with 6989 nodes and max depth of 32.

**Fig 8: (1st) Images from original dataset, (2nd) Example of web cam test data, (3rd) Example of second test data.**

With this updated preprocessing approach, we wanted to test our CNN model to see if we had been successful in creating a more flexible model to real world imaging conditions. To accomplish this, we introduced a completely unseen dataset from the training data. This second dataset seen in the third example in figure 8 is much more diverse, and differs in size, skin tone, image background, and image resolution.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Accuracy (%) on dataset**  **Before preprocessing** | **Accuracy on dataset(%)**  **After preprocessing** |
| **original dataset** | 99.8 | 92.88 |
| **team members webcam** | 18.4 | 68.39 |
| **second dataset with no additional training** | N/A | **93.08** |

**Table 4: Table of model generalizability experiments**

Excitingly, our model was still able to achieve over 93% accuracy on this completely unseen data by transforming these new images into the same preprocessed domain for classification. This offered support that our approach could begin to generalize beyond its initial training dataset and provide a viable means of achieving our long-term project goals of supporting real-world ASL hand sign recognition.

**FUTURE WORK**

Overall, we believe we have found a very promising path forward for automated transcription of ASL. Immediate future would be to adapt our image processing approach to video processing. This would allow for classification of the missing two dynamic ASL letters J and Z. Additionally, by using a staged classifier approach, full sentence recognition could be explored and potentially achieved with these video inputs. Taking this the final step would require the incorporation of facial recognition to include emotional context and infer sentence meaning. This could feasibly provide a full ASL recognitional capability and would be extremely applicable to other machine gesture recognition use cases.

**REFERENCES**

Akash, N. (2018). ASL Alphabet [Data set]. *Kaggle.* <https://doi.org/10.34740/KAGGLE/DSV/29550>

Owen, F. (2022)Synthetic ASL Alphabet[Data set]. *Kaggle.* <https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet>

Lanang A. (2021) CNN with Media pipeline for Sign Language Recognition. *Kaggle.* <https://www.kaggle.com/code/mlanangafkaar/cnn-with-mediapipe-for-sign-language-recognition/>

Bengaluru, Karnataka. (2019) Resampling methods using Bootstrap & CV. *Kaggle.*

https://www.kaggle.com/code/sasha18/resampling-methods-using-bootstrap-cv/notebook

Prateek Joshi. (2020) Bootstrap Sampling in Statistics and Machine Learning. *Kaggle*

https://www.analyticsvidhya.com/blog/2020/02/what-is-bootstrap-sampling-in-statistics-and-machine-learning/

[Daniele Berto.](https://www.kaggle.com/danready) (2020) Naive Baysian Algorithms. *Kaggle*

<https://www.kaggle.com/code/danready/bertodaniele-mnistsign-bayes>

[Hieu Tran](https://www.kaggle.com/heraclex12) (2020) HOG and Logistic Regression. *Kaggle*

<https://www.kaggle.com/code/heraclex12/hog-and-logistic-regression-achieved-87>

Aman, P. (2022, March 1) Hand landmarks detection on an image using Mediapipe. *Analyticsvidhya.* <https://www.analyticsvidhya.com/blog/2022/03/hand-landmarks-detection-on-an-image-using-mediapipe/>

[Simon Kiruri](https://www.section.io/engineering-education/authors/simon-kiruri/) (2022, February 11) Creating a Hand Tracking Module using Python, OpenCv, and MediaPipe. *Section.* <https://www.section.io/engineering-education/creating-a-hand-tracking-module/>

Rishabh, S. (2019, May 6) Python | Thresholding techniques using OpenCV | Set-2 (Adaptive Thresholding)*. Geeksforgeeks.* <https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-2-adaptive-thresholding/>

Raj (2021) Python | An exhaustive guide to decision tree classification in python 3.X | Set-2 (Decision Tree)*. TowardsDataScience*. <https://towardsdatascience.com/an-exhaustive-guide-to-classification-using-decision-trees-8d472e77223f>

Build software better, together. (n.d.). GitHub. <https://github.com/pooji0401/Image-Classification-System-using-Decision-Trees/blob/master/Image+Classification+System.ipynb>