ECGR 4106 Final Project - American Sign Language Detection

Abrar Altaay, Tyler Chambers, Cole Fredrick, Jaiden Hausler

***Abstract*—Deep learning models that perform object detection are proposed in this paper to identify the alphabet hand signs of American Sign Language (ASL). This paper goes over the image data set of ASL hand signs, the training process using ResNet-18, DenseNet-161, and YOLO-v4 models, and their results. Test sets were used to validate the models and results are provided to show the effectiveness of the three models. The hand sign detection is tested through images and videos and showed high accuracy's. Using this machine learning approach allowed for a practical ASL alphabet translation. — GitHub Repository: https://github.com/tchamb10/Final_Project_RealTime**

I. Introduction

Object detection and classification is a technique used to identify objects in images or videos that typically rely on deep learning to produce meaningful results. When humans look at images or videos, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence of humans. There is an infinite number of applications that one can apply object classifications algorithms that have the potential to increase accessibility, enhance automation, and improve human interactivity. An example where deep learning can improve quality of life is easing translation of sign languages to better communicate with others.

Sign language is a language that uses hand gestures and facial expressions to communicate. These signs mean different things to different people, so it can be difficult to know what the signer is saying or what type of sign language they are speaking. It is commonly used by people who are deaf or have a hearing impairment. This project will strictly focus on the American Sign Language alphabet as ASL is one of the most popular sign languages spoken. [1]

Currently about 500,000 Americans use the American Sign Language (ASL) in the United States. There is a difficult language barrier between ASL users and English speakers, as ASL is not well known by those who are not deaf or hard of hearing. According the United Nations statistics, roughly 70 million deaf people worldwide use a sign language as their first language. Using a neural network to train on thousands of images of hand signs, we can train a network to classify what hand sign is present in real time. The impact of having a live a translation can lessen the language barrier that is currently present. [2] [3]

With the goal of providing more accessibility, the team tested with three different neural models and tested each one for the highest accuracy. The three networks used were ResNet-18, DenseNet-161, and YOLO-v4.

II. Data Set

The first data set that was used in this project was collected from a popular database website known as Kaggle.com authored by Debashis Sau. The data set was a collection of images of the alphabet gestures from the American Sign Language separated in 29 folders that represent each letter, a delete gesture, a space gesture, and a nothing folder. In total there are 223,075 raw RGB images in the data set spilt into training and validation with a 80% and 20% split respectively. 178,459 images in the training set, and 44,615 images for the validation set with each image being 640x480 PP resolution. This data set was used to train a ResNet-18 model and a DenseNet-161 model for object classification. The link to the data set can be found in the reference table VI.

Another data set was used from Roboflow.com authored by David Lee and this set was used to train a YOLO-v4 model, this data set included 1,728 raw RGB images that are 416x416 resolution that include bounding boxes. The set was split into 3 sets, training, validation, and testing with the training taking 95% of the images, the validation taking 3% and the testing taking 2%. The total values for each split is 4500, 144, and 72 images respectively. The data set included a .TXT file for each image that defined the bounding boxes and label that is then fed into the YOLO-v4 model. Using data augmentation, the group was able to increase the data set to around 5,000 images.

III. Training Models

Before training can begin, the data must be loaded into the script. The team used a torch function called ImageFolder to load the data and complete pre-processing transformations such as resizing and normalization. The data was then separated into Training and Validation splits of 80% and 20%. No further augmentation of the data was done on the two classification models, however, that is not the case with YOLO-v4. The team took a different approach with YOLO-v4, the data was loaded into RoboFlow.com and slight rotations were made to each image to expand the data-set. The data was then imported into the YOLO python notebook through a personal RoboFlow.com passcode specifically for our data-set. Once the dataset was loaded, the data config file was created with the images and labels loaded in, this config file is used during the training. Code from RoboFlow.com was used to help speed up training and efficiency specifically for google colab.

It is imperative to test multiple training models to identify the most accurate and best suited model for the data set. The

team utilized Google Colab GPU to train all 3 different neural network models for this project. The first model that was approached was ResNet-18 model that included 11,174,000 parameters. ResNet was used because of the relatively faster training times as compared to DenseNet-161. Before training, the data was first resized from 640x480 to 64x64 for each image to reduce training time length, but keeping enough data for accurate training. The images first then needed to be normalized using the mean and standard deviation of the entire data set. After that is complete, the model was then able to be trained for 10 epochs with a batch size of 256. This batch size was used to help maintain faster training times, while also optimizing for GPU memory issues that were encountered with higher batch size numbers.

To perform comparative analysis against ResNet-18, a DenseNet-161 architecture was implemented. The amount of parameters that this module carries is roughly 28.7 million, which is over double that of the ResNet-18 implementation. DenseNet overall allows for increased capacitance for model complexity but in turn increases the amount of resources needed from the systems GPU for computations. The transformations performed for this architecture on the data-set was a resizing of images from 620x480 to 64 by 64, including a normalization using the mean and standard deviation of the data-set.

TABLE I
MEAN AND STANDARD DEVIATION FOR RESNET-18 AND DENSENET-161

	ResNet-18	DenseNet-161
Mean	0.5404, 0.5179, 0.5214	0.5404, 0.5179, 0.5214
STD	0.1834, 0.2156, 0.2281	0.1834, 0.2156, 0.2281

The final goal for the team was to localize and classify hand movements for ASL translation. Localizing the target image and then classifying will allow for even better classification accuracy. To do this, YOLO-v4 which is based on the DarkNet Architecture was used to complete this goal. YOLO-v4 carries over 60 million parameters that are spread out over 160 layers. These layers include batch normalization and max pooling. The model was trained on 2125 iterations at a batch size of 64. The training session took around 2 hours and 30 minutes and resulted in great accuracy as shown below in the Results section.

IV. RESULTS

When testing ResNet-18 the model yielded an a 99% validation accuracy with a 97% test accuracy at a 10 epoch iteration count. A personal test set of images of different hand gestures was created and uploaded to the trained network to test the accuracy of the model. For example, the group uploaded a image of the hand sign 'O' and the model was able to successfully predict the letter. Refer to figure 1.

The confusion shows the actual vs predicted values for every letter in the data set, including space, delete, and nothing, which validates the accuracy across the entire data set. Refer to figure 2.

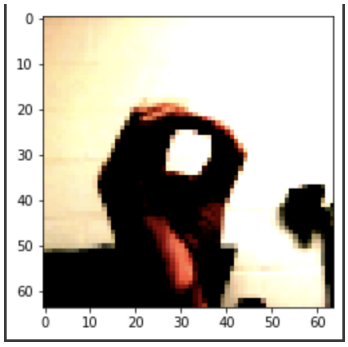

Fig. 1. ResNet-18 user inputted sign for the letter 'O'

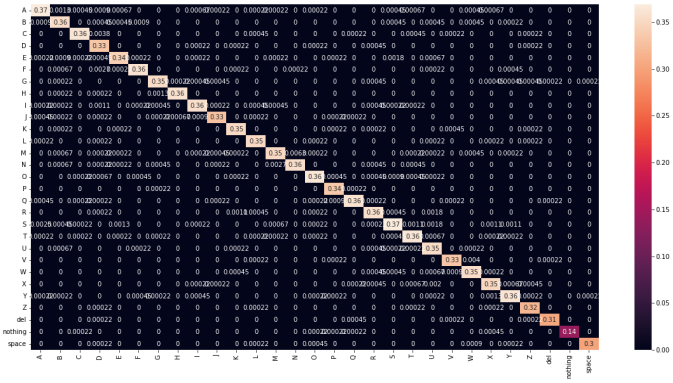

Fig. 2. ResNet-18 Confusion Matrix Results

After DenseNet-161 was then trained and verified, yielding a 99% validation accuracy, and a 98% test accuracy. This is a one point difference between ResNet's test accuracy. Again, the team uploaded a custom image of a hand sign, to the network to test the prediction accuracy. For this example, the group tested with the letter 'C', where again the group uploaded an image of the letter sign to the trained network and the model successfully predicted the image to be the letter 'C'. Refer to figure 3.

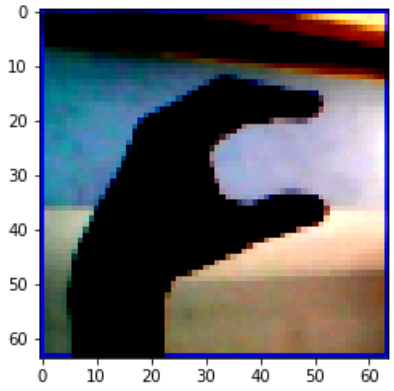

Fig. 3. DensNet-161 user inputted sign for the letter 'C'

The confusion matrix again shows the actual vs predicted values for every letter in the data set which validates the accuracy across the entire data set. Refer to figure 4.

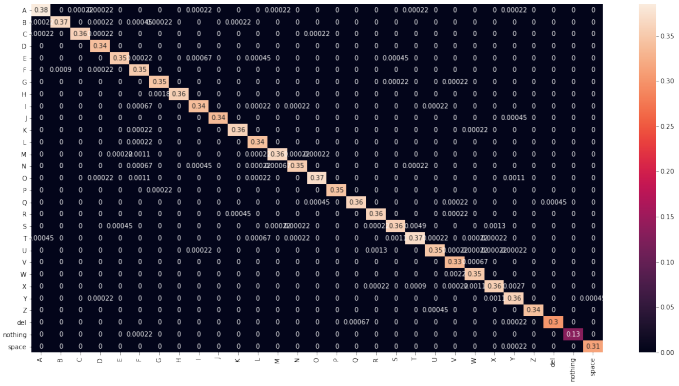

Fig. 4. DensNet-161 Confusion Matrix Results

The last model that was tested was YOLO-v4 for real time applications. The model output a 93% test accuracy with the video being processed at 30 frames per second on average. The group created several videos to test the model that included someone gesturing the alphabet in ASL. While testing, the group noticed that videos that did not include a users face or body performed better. Currently, the model has issues identifying 4 letters M, N, S, and T. This is because the gestures for these letters all look extremely alike between one another, the difference between the letters is the placement of the thumb when making a fist. For example of a bounding box, refer to figure 5, the model was able to draw a bounding box around the hand gesture. The team uploaded a short 1 minute video into the trained YOLO-v4 model to classify different letters of ASL. The following link will show the results of the video. https://drive.google.com/drive/folders/1ne3EoMYwUbH3jSYPkXcpb9VCz8EGET3v?usp=sharing.

In the video, one can see the model is able to successfully draw bounding boxes around the hand gestures and accurately predict the letter the user was gesturing to a certain accuracy. At times, the model will draw multiple bounding boxes for a certain gesture, again this is because those letters might look especially similar to another hand gesture.

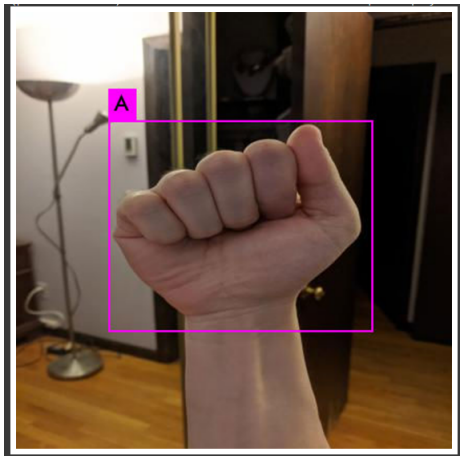

Fig. 5. YOLO-v4 bounding box prediction around the letter A

V. Conclusion

Deep learning networks like ResNet-18, DenseNet-161, and YOLO-v4 can be used to develop communication applications for non-ASL users. This alleviates the language barrier and improves accessibility for people who are hard of hearing or deaf.

Comparing ResNet-18 to DenseNet-161, their accuracy's were very similar. The two models both had 99% validation accuracy. DenseNet-161 had a 1% higher test accuracy than ResNet-18. The training time differed significantly with ResNet-18 taking less time by a total of 2 hours and 3 minutes. Therefore, ResNet-18 is preferred for optimization because it takes less computational power. Also, ResNet-18 requires less storage and would be the best option for deployment.

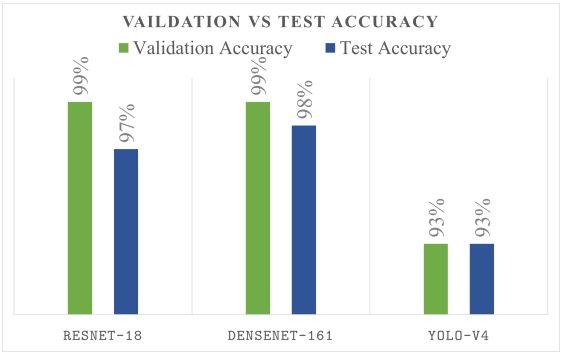

Fig. 6. Validation vs Test Accuracy graph for each model

YOLO-v4 gave a lower percentage than ResNet-18 and DenseNet-161 with a test accuracy of 93%. However, YOLO-v4 is not comparable because it exceeds in functionality making it's accuracy more desirable. A problem was experienced with YOLO-v4 having trouble deciphering between the letters m, n, s, and t because of the thumb placement in ASL. Adding more images to the data set would allow YOLO-v4 to learn key distinctions between these letters. The remaining letters in the ASL alphabet were efficiently recognized, With this model preforming ASL alphabet detection on videos, this is a promising way for non-ASL users to translate ASL.

VI. References

1) "American Sign Language Letters Object Detection Dataset," Roboflow. https://public.roboflow.com/object-detection/american-sign-language-letters
2) American Sign Language Alphabet Object Detection Dataset, www.kaggle.com, https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset.
3) M. I. left my 'comfortable' job on F. to fulfill my dream-educating people worldwide, "How Many People Use Sign Language Worldwide?" https://www.ilovelanguages.com/how-many-people-use-sign-language-worldwide/
4) Github Repository: https://github.com/tchamb10/Final_Project_RealTime

5) YOLO-v4 video: https://drive.google.com/drive/folders/1ne3EoMYwUbH3jSYPkXcpb9VCz8EGET3v?usp=sharing