American Sign Language Handshape Detection

December 18, 2018

Victor Geislinger

1 Capstone Project

1.1 Definition

1.1.1 Project Overview

American Sign Language (ASL) is a sign language that does not use speech to communicate and is mostly used by the American Deaf population. Though used throughout the English-speaking United States, it is in fact its own language seperate from English and relies on building the language's syntax with multiple visuals such as handshapes, movements, positions, and nonmanual markers. Although there are many variations of sign language specific to different languages, regions, and needs, being able to use a computer to detect ASL would be extremely useful in not only ASL translation but also other sign language translations as well as non-language geusture recognition.

This project focuses on classifying the handshapes that represent the letters in the English alphabet using still images using deep learning image recognition techniques. The focus on still images will allow a start for full ASL translation which would require aspects to be measured like handshape position (handshapes in different positions and orientations can affect meaning) and movement (in ASL, meaning has a strong tie to time dependence, like all languages) that can be measured in other types of datasets like video.

It should be noted the handshapes being classified are not all the handshapes used in ASL (such as handshapes associated with numbers) and signs that represent letters but require movement have not been included, specidically "J" and "Z" handshapes. Lastly, though this project will consider handshapes in the position associated with their associated letter, the letters "P" and "Q" are in fact the same handshapes of the letters "K" and "G" respectively but in different positions. Since the still images are significantly different between these related letters, it was deemed appropiate to consider these pairs as separate classifications.

1.1.2 Problem Statement

The project's goal is to classify static images of the ASL handshapes associated with 24 English letters "A"-"I", "K"-"Y". Since the dataset will consist of static images, image recognition strategies using deep learning will be used. The dataset used will be from a paper that had a similar goal and will be used as an overall benchmark to evaluate this projects performance. The subgoal is to classify the handshapes with better accuracy than the paper's model which used both the static images and depth sensing information.

The dataset will be split into training, validation, and testing subsets and multiple image recognition models will be trained and compared. The subsets will be consistent accross the different models so that these models can be more easily compared. It will be disscussed later in this paper but it should be noted now that five different subjects provided the handshape images; the trainig and validation sets will be randomly made from four of the subjects and the fifth subject's images reserved for the testing set.

1.1.3 Metrics

This project evaluates the performance of a given model by observing the number of correct handshapes it classfies $accuracy = \frac{N_{correct}}{N_{predicted}}$ where $N_{correct}$ is the number of correct predictions and $N_{predicted}$ is the total number of predictions made. Another metric used to represent the precision and recall of the model is the F_1 score given by $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. This project uses a F_1 score over a different F_β ($\beta \neq 1$) score since both precision and accuracy should be considered with approximately the same weight (we care about classifying the correct hanshape as much as we care about misclassifying a letter).

1.2 Analysis

1.2.1 Data Exploration

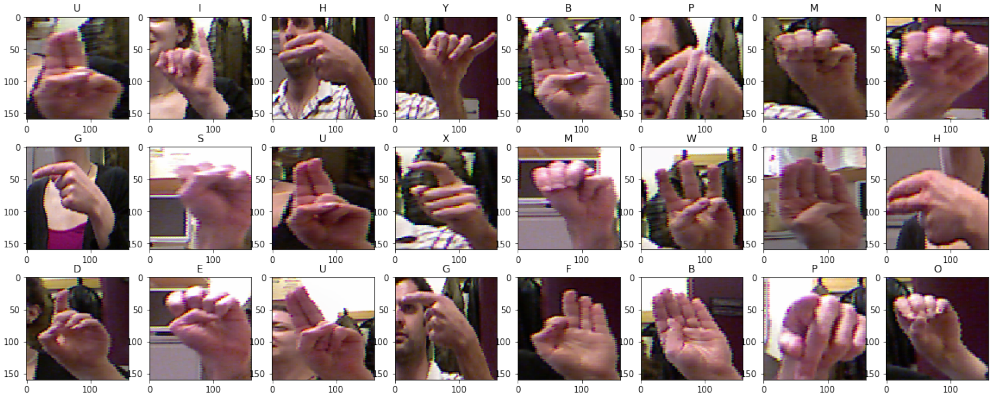
The dataset used was the ASL FingerSpelling Dataset from the University of Surrey's Center for Vision, Speech and Signal Processing Section 2 Section 2. This dataset contains both colored images and depth sensing data collected from a Microsoft Kinnect. (Note that this project did not be use the depth sensing data since the project focused on using static images.) The images are in color and include 24 different handshapes each representing a letter from the English alphabet; "J" and "Z" are excluded since these letters are dependent on movement and therefore don't have a static image representation.

The dataset images have been cropped around the handshape though each cropping results in a differently sized image fitting within a 275 pixels by 250 pixels window with a resolution of 72 pixels per inch. The background behind the handshape is not uniform or consistent. The dataset contains approximately $65,000$ images generated by five different non-native ASL signers with over 500 samples of each of the 24 different handshapes. The handshapes in the image feature some rotational differences as the subjects were instructed to adjust hand position for the camera. These positional adjustments however still preserve the common position associated with the English letter it represents.

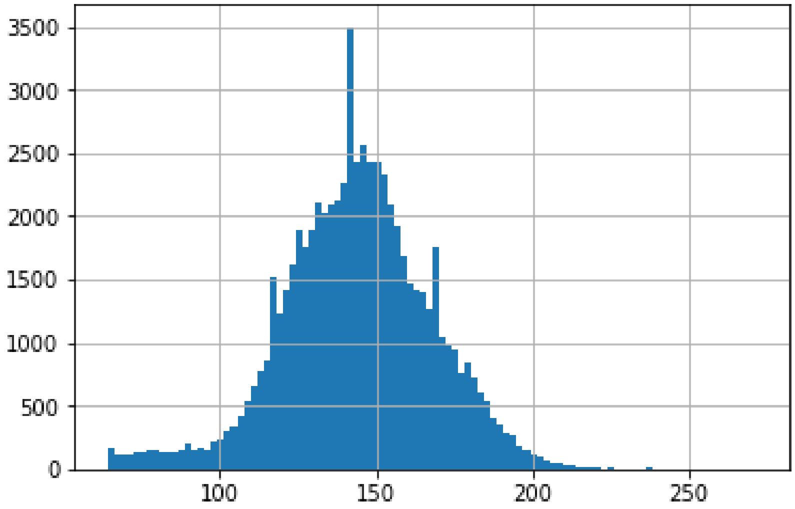
1.2.2 Exploratory Visualization

Exploring the dataset, it can be observed that the images have different dimensions though all fitting within a 275 pixels by 250 pixels window. This makes sense in the data collection since the handshapes have varrying aspect ratios. The images' heights can be observed as being roughly a normal curve centered at approximately 140 pixels. The images' widths however have a right skewed distribution. The width distribution implies that most handshapes are relatively narrow likely coming from the fact that most handshapes of English letters have the hand positioned with the fingers perpindicular to the horizon.

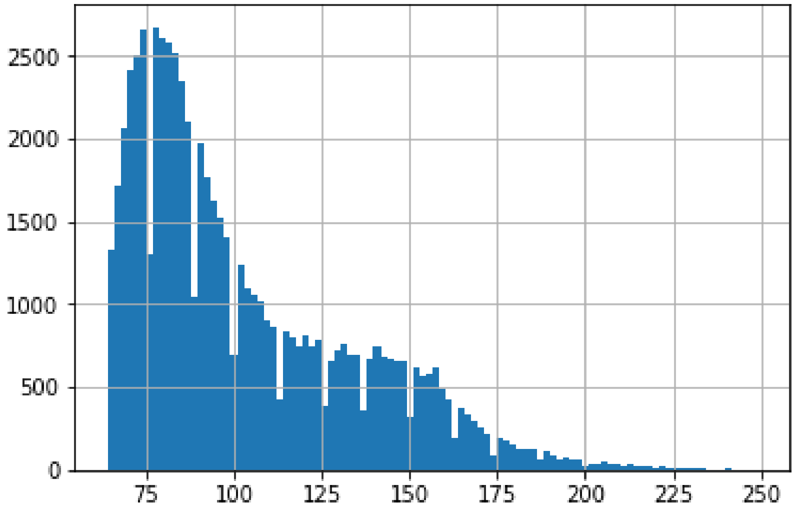
Investigating further into the aspect ratio of the images and the area (number of pixels in an image), a right skewed distribution is observed for both. The area gives us literally how many



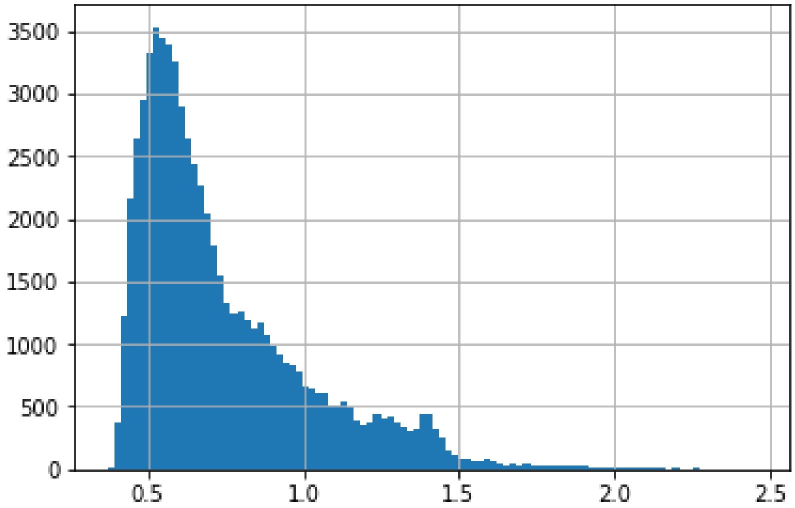
Random sample of images from the dataset.



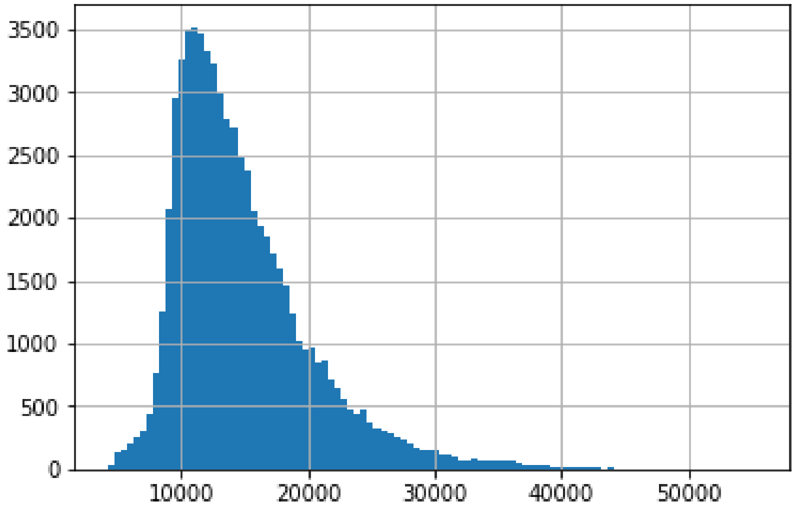
Histogram of images' heights in pixels.



Histogram of images' widths in pixels.



Histogram of images' aspect ratio.



Histogram of images' area (total number of pixels in image).

inputs there would be for the model if we simply input the pixels and directly affects the image size; larger areas/more pixels means large file sizes and more data. It can also be observed that over 75% of images have a ratio smaller than 1.0 meaning that a large majority of images are taller than they are wide.

Looking at our images' different sizes, one can pick up on some patterns in the data. The smallest height and width are 64 pixels, while the maximums for height and width are approximately 270 and 250 pixels respectively. One can notice that the aspect ratio trends below 1.0 meaning that most images are taller than they are wide. It can also be observed that the area metric is heavily skewed toward smaller values. This fits with the aspect ratio and likely is because of the skew in width. The average number of pixels in the images is about 15000 pixels and less than 25% have more than 18000 pixels.

1.2.3 Algorithms and Techniques

Since this project is focused on classifying static images it was determined that deep learning techniques, specifically utilizing convolutional neural networks (CNNs), would be the most relevant and effective method for this situation. However, since CNNs require all images to be the same dimensions it was determined the best method for this project was to scale the images to the same size. This will be discussed further in the "Data Preprocessing" section but in summary it was determined to scale all the images to 160 pixels by 160 pixels to preserve the relative shape of the majority of images.

Multiple model architectures were used and compared. A basic CNN model was built from scratch to determine the accuracy that could be achieved before going into more advanced techniques. Afterwards transfer learning was used to improve accuracy. VGG-16, VGG-19, and ResNet-50 architectures were used with bottleneck features to help with speed. These models results were then compared amongst one another in terms of the overall accuracy and F_1 score as well the accuracy in correctly classifying specific handshapes.

1.2.4 Benchmark

The first and simplest benchmark will be comparing the developed model with a "random choice" model. With each handshape being equally likely, we would expect the "random choice" model to only identify handshape instances $\frac{1}{24} \approx 4.2\%$ of the time on average.

A goal of this project is to achieve better performance than if specialized equipment other than a camera to take images were to be used (like the Microsoft Kinnect). We can then use the performance of the *"Spelling It Out"* paper's random forest model as our idealized benchmark. Observing the paper's confusion matrix which used four of the subjects to train and validate the model and one subject's images to test performance, similar handshapes were misclassified with the handshapes least correctly identified being "T", "O", "S", and "M" (7%, 13%, 17%, and 17% of the time correctly identified respectively). The handshapes that were identified most accurately were "L", "V", "B", and "G" (87%, 87%, 83%, and 80% of the time correctly identified respectively). The paper's model achieved an overall mean precision of 73% and 75% using only the images and using both images and depth data respectively.

1.3 Methodology

1.3.1 Data Preprocessing

Since it was determined that CNNs were to be used (as discussed in the previous section), the images needed to be scaled to the same dimension. It would be ideal to keep as much information as possible in resizing the images. The extreme solution would be to scale the images to the maximum height & width (272 pixels & 249 pixels respectively) however this would create larger file sizes and could greatly distort the majority of the images. Therefore, in resizing the images most images should retain most (if not all) of their information while there are fewer images losing information. Note that the images should not be cropped since they have already been cropped around the images' handshapes.

It was observed that most images are taller than they are wide. It then made sense that resizing the data should favor taller images since this is more common in the data. One solution was to add padding to images so that the width matches with the height. However, adding this padding could give bias to the model if particular handshapes have differing aspect ratios. In other words, the model could simply learn based on the padding instead of the handshape in the image. Thus stretching/squeezing or rescaling the image seems to be preferred even if the images will lose more information.

After determining that rescaling was the best strategy, it needed to be determined how large the rescaled images would be for the CNN architecture. It has already been said that a square image was preferred. More than 75% of the images have aspect ratios less than 1.0 so most images could simply be rescaled to a square by changing the width.

Restricting the width to 125 pixels would keep information preserved for about 75% of the images. This seems to be reasonable for width but height has to be considered as well. Restricting the image to 125 pixels for height would mean information would be preserved for less than 25% of the images. This appeared to be a poor tradeoff especially considering that most images were tall and therefore most pictures would lose information.

Focusing on height instead, a restriction of 160 pixels would preserve all the height information for about 75% of the images. It was observed over 75% if the images would preserve all its width information. This was great since most images would preserve the relative shape and pixel density for the vast majority.

In summary, the images were resized to 160 pixels by 160 pixels which preserved most of the images' information. The size was also ideal since each dimension was divisible by 32 giving a decently sized power of 2. The total number of pixels for each image after resizing was 25600 total pixels which allowed for a large proportion of the images to not lose information after scaling.

1.3.2 Implementation

The first goal was to build a CNN from scratch (with no transfer learning). (Note all work for the following sections can be found in notebook titled *asl_recognition.ipynb* unless stated otherwise). The first step was to identify all of a subject's images to be the testing data set with the rest of the images from the other four users would be used for training and validation. This was to emulate our benchmark from the paper referred to earlier. The training and validation sets were then randomly split from the remaining so that 80% of the images were for training and the other20% identified for validation. These testing, training, and validation image sets would be used consistently for any model developed.

The next step was to prepare the model. Originally the goal for the basic model from scratch was to use colored images (RGB) as input. However, having three channels made it difficult to load the training and validation sets into memory as it was too memory intensive. So the basic model used grayscale (one channel) images as input.

The images were also scaled to 224 pixels by 224 pixels to make it easier to compare the ResNet50 transfer learning model to be used later. It was predicted that this model would do best against the other planned transfer learning models (VGG16 and VGG19). The grayscale and scaled images for testing and validation were then loaded into memory as tensors so they could be applied to the model to find the proper weights.

The basic model was created with three convolution layers using the reLU activation function and with 16, 32, and 64 filters. Each of these layers were followed by a max pooling layer with a pooling size of 2. The model was finished with a dense layer using the softmax function of 24 outputs (for each handshape to predict). Below is the code used in the notebook to define the model:

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation='relu',
                 input_shape=(224, 224, 1)))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(GlobalAveragePooling2D())
model.add(Dense(24, activation='softmax'))
```

Next the model was trained using the training data and validation data. The model ran with 16 epochs with batch sizes of 20 images. This produced weights for the model that could then be used to evaluate the model.

The model was then evaluated using the testing set images after scaling them in the same grayscale and 224 pixels by 224 pixels (also loaded into memory as a tensor). The trained model produced about a 43% overall accuracy. It was determined that the next step for improvement would be to use the technique of transfer learning.

1.3.3 Refinement

Using the model built from scratch, training took a relatively long time even with small batches and few epochs. It was determined that using transfer learning would be an effective use of computing resources to help improve the model's overall accuracy. Attempts were made using the models of VGG16, VGG19, and ResNet-50.

The next attempt was to use transfer learning via the VGG16 model architecture. It was decided to use the technique of bottleneck features to speed the training with the model. Thus the VGG16 model was loaded and the final classification layer was removed to calculate bottleneck features. The training images were then fed into the neural network to create the bottleneck features. These features could then be trained on using a CNN (similar to the basic model) to predict on the relevant data.

Various CNNs and scaled images were tested with the VGG16 model. RGB images were used and the pixel density ranged from 80 pixels by 80 pixels to 160 pixels by 160 pixels. Many variations on the CNN were made but mostly included no more than three convolutional layers. Overall accuracy ranged from about 50% to about 65%. The best overall accuracy achieved by VGG16 was about 66% using three convolutional layers of 256, 128, and 32 filters with a batch size of 9000 and 400 epochs.

The next attempt was with a new model architecture, VGG19. A similar procedure to the VGG16 model was used with this architecture; transfer learning was implimented and bottleneck features were used to help increase the speed of training. However after variations with this model, the overall accuracy was still unable to break the best overall accuracy of 66% achieved by the VGG16 architecture.

The final set of attempts were done with the ResNet-50 architecutre (also transfer learning). This model was able to achieve an overall accuracy slightly higher than the previous models. Again bottleneck features were used but this time using 224 pixels by 224 pixels RGB images. An overall accuracy of about 70% was achieved using a CNN with 3 convolutional layers (256, 12, and 32 filters) on the bottleneck features, a batch size of 512, and 2048 epochs. This seemed to be the level that could achieved with reasonable access to computing resources.

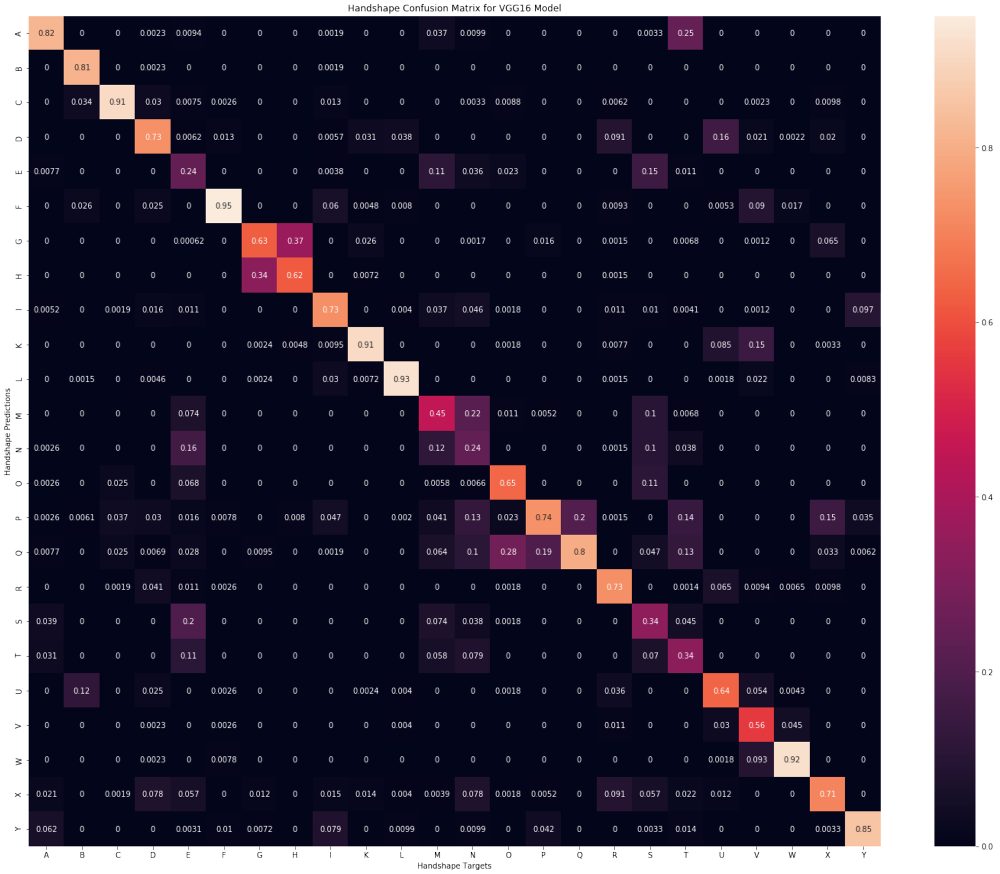
Overall it seemed that the least accurate model was, as expected, the "from-scracth" model with the best score of about 43%. The next set of improvement came from VGG16/VGG19 where the models were able to achieve significantly better results with the best overall accuracy of about 66%. There was further improvement when ResNet-50 was used but in this case, the improvement was not nearly as dramatic with the best overall accuracy only slightly more than the VGG16 at about 70%.

1.4 Results

1.4.1 Model Evaluation and Validation

The final model chosen was the ResNet-50 model with an overall accuracy of about 70%. This model used transfer learning and used bottleneck features to accelerate the training speed. The model was fed RGB images scaled to 224 pixels by 224 pixels. The neural network trained with the bottleneck features used 3 convolutional layers (256, 12, and 32 filters), a batch size of 512, and 2048 epochs. Below is the code used for the neural network on the bottleneck features:

```
model = Sequential()

model.add(Conv2D(filters=256, kernel_size=1, input_shape=features.shape[1:]))
model.add(Dropout(0.4))

model.add(Conv2D(filters=128, kernel_size=1))

model.add(Conv2D(filters=32, kernel_size=1))
model.add(Dropout(0.3))

model.add(GlobalAveragePooling2D())
model.add(Dropout(0.3))

model.add(Dense(24, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
```

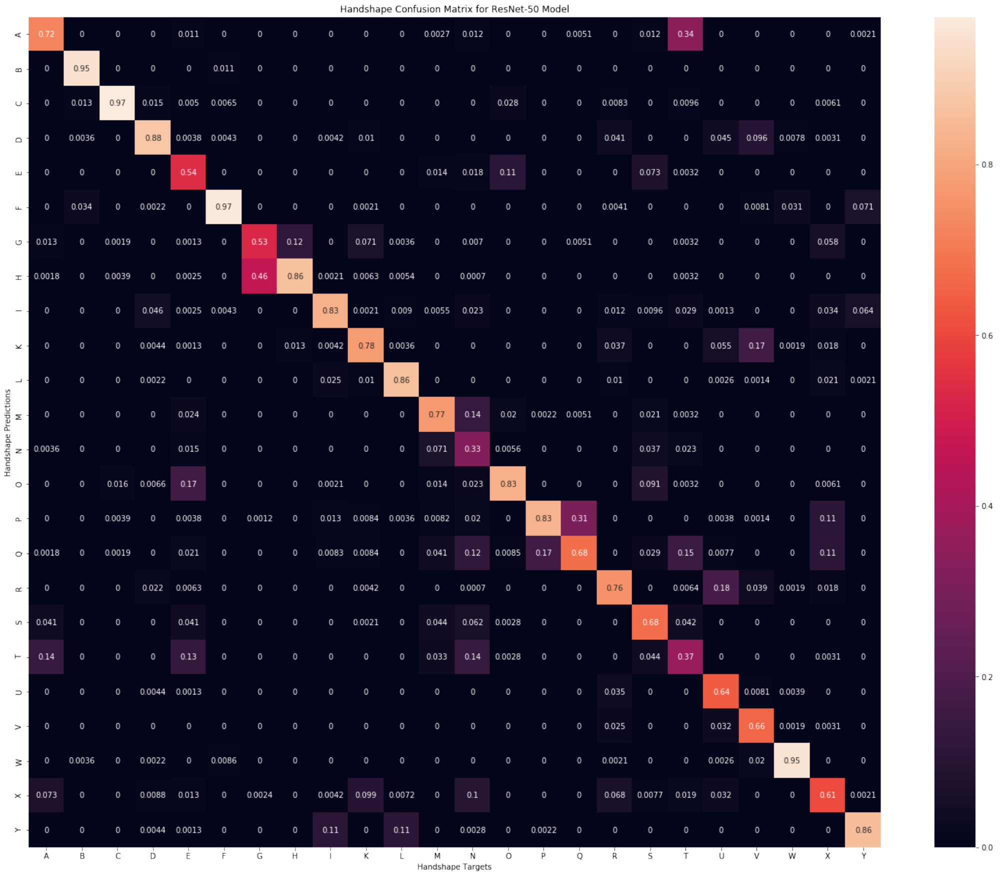


Confusion matrix of the VGG16 model's results; handshape targets in the x-axis and predictions in the y-axis.

This model was chosen over the other candidate models based not only it having the best overall accuracy, but also because on how it made mistakes for particular handshapes. Observing the confusion matrices (see figures below) between the ResNet-50 model and the VGG16 model (the runner-up candidate with an overall accuracy of 66%), it can be observed that there are more frequent incorrect predictions from the VGG16 model. Or in other words, when the VGG16 model predicts incorrectly there tends to be more handshapes it predicts the target as compared to the ResNet-50 model.

It should be noted that the ResNet-50 model does perform worse in identifying particular handshapes; VGG16 predicts the handshape "A" targets correctly 82% of the time compared to ResNet-50's 72%. However, the ResNet-50 model has significant improvements in other traditionally difficult handshapes. Such examples are the "M" and "S" handshapes which only had 45% and 34% of their respective targets predicted correctly from VGG16 compared to ResNet-50 achieving 77% and 68% respectively.

This seems to be a common pattern between the two models with particular similar looking handshapes where ResNet-50 has more correct predictions. Handshapes like "A", "M", "N", "T", and "S" are similar looking as they all involve a closed fist with slight variations on the thumb placement. These handshapes are already difficult for human ASL learners to distinguish, so the fact that the ResNet-50 model does better in distinguishing these handshapes gives an edge over the VGG16 model. Sacirficing some accuracy in some already more readily distinguishable



Confusion matrix of the ResNet-50 model's results; handshape targets in the x-axis and predictions in the y-axis.

handshapes to greatly improve the identification of more easily consufable handshapes is overall more desirable.

1.4.2 Justification

After determining that the Resnet-50 model was the best final model amongst the trained models, the benchmark must be compared to it. We can first notice that our overall mean percision for our model was about 70% which was lower compared to the benchmark paper's claim of 73% and 75% for using just RGB images and RGB images with depth sensing respectively.

However, as we will see later, this seems unlikely since the confusion matrix in the paper (using the results from the combined image and depth data model) will overall underperform in comparison to our model. Since the paper does not provide the raw data on how the mean percision and/or the confusion matrice was created we can only speculate this discrepency. Perhaps there were more letters in their testing set meaning having a higher accuracy and percision for one handshape greatly affected the overall metric. This seems unlikely as the paper directly stated the reasearchers used the fifth subject's data for testing, whom was also used in our testing.

Since it cannot be confirmed how the percision was determined, we will use the confusion matrix as our comparison because it has more individual detail for each handshape. Note however that this confusion matrix provided did the unorthodox adjustment by reporting the percent of the targets in each predicted class. This is fine as we compare the performance of each handshape with the performance by the ResNet-50 model. However, we cannot calculate other metrics like recall. Again, this is a minor issue as the goal is to compare the benchmark model with the ResNet-50 model.

When comparing the two models, we can see that the ResNet-50 has higher precision on it's best predicted handshapes than the benchmark's best predicted handshapes. The benchmark best performed on the handshapes "L", "V", "B", and "G" with 87%, 87%, 83%, and 80% of the targets correctly predicted respectively. However the ResNet model best performed on the handshapes "C", "F", "B", and "W" 0.97%, 0.97%, 0.95% and only then dipping below 90% with "D" at 88%. In fact, almost half of all the handshapes ("B", "C", "D", "F", "H", "I", "L", "O", "P", "W", and "Y") were above 80% compared to those four handshapes with the benchmark model. This demonstrates that the ResNet-50 model has significant improvements overall in handshape recognition.

It should be pointed out that although the ResNet m

east correctly identified being "T", "O", "S", and "M" (7%, 13%, 17%, and 17% of the time correctly identified respectively). The handshapes that were identified most accurately were "L", "V", "B", and "G" (87%, 87%, 83%, and 80%

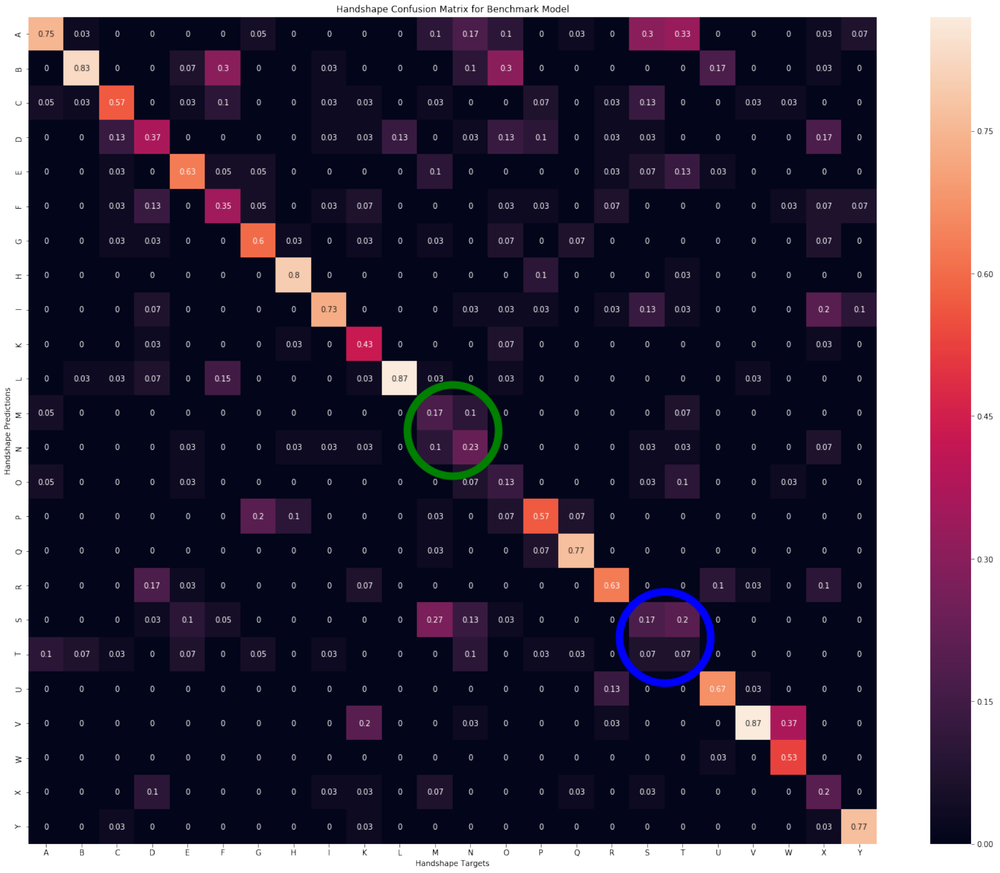
1.5 Conclusion

1.5.1 Free-Form Visualization

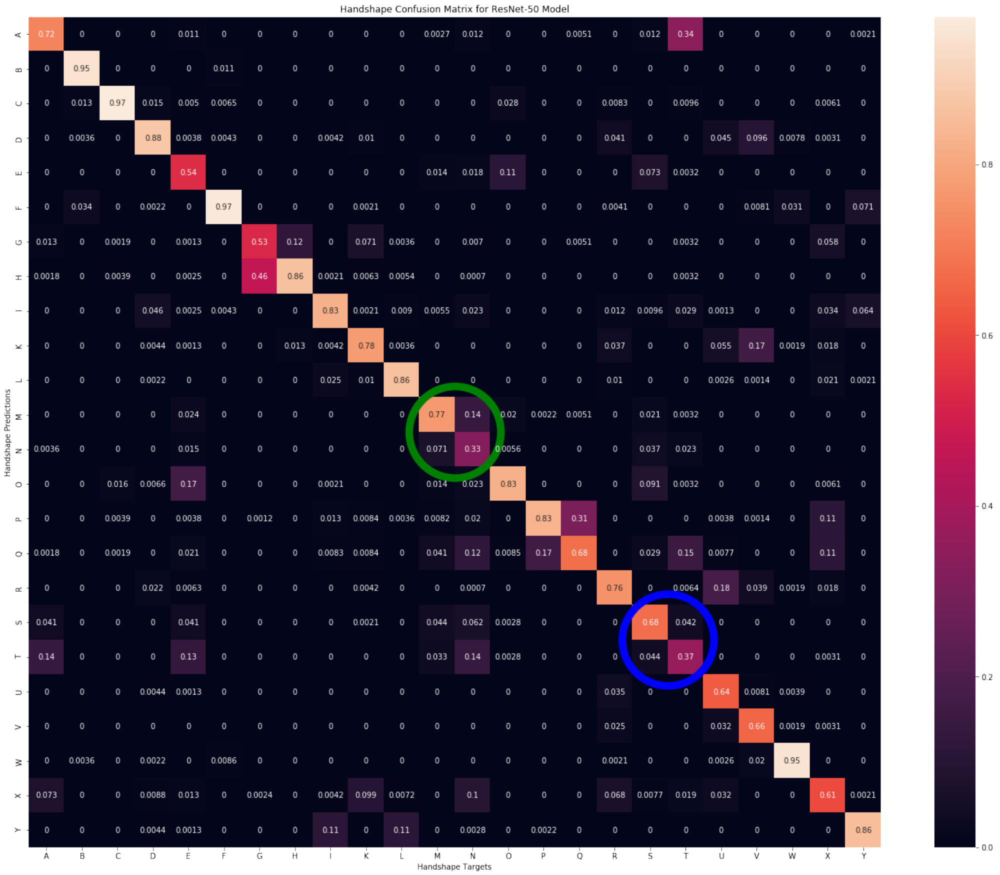
A huge achievement of this project that the trained final model (using ResNet-50 architecture) was not only better overall at predicting hanshapes compared to the benchmark model, but also doing significantly better in predicting similar looking handshapes. This is most ephasized for the pairs "M" & "N" and "S" & "T".

The differences between the two confusion matrices has already been thoroughly explored but here we highlight the two pairs with a green circle ("M" & "N") and a blue cicle ("S" & "T").

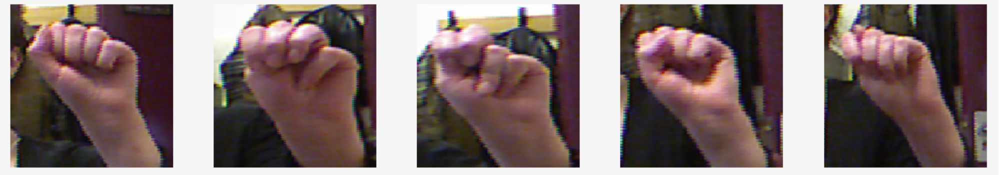
It can be observed in the benchmark's confusion matrix that these pairs are commonly predicted as it's similar conterpart. The ResNet-50 model does have some mistakes in prediction but



Confusion matrix of the benchmark model's results with the pairs "M" & "N" and "S" & "T" emphasized with a green and a blue circles respectively.



Confusion matrix of the ResNet-50 model's results with the pairs "M" & "N" and "S" & "T" emphasized with a green and a blue circles respectively.



The handshapes (from left to right) "A", "M", "N", "S", and "T" all involve a closed fist and different thumb placement relative to the fist

does significantly better. It should be noted that if one were to observe along the vertical axis of the handshapes (all the predictions for that given target), one may notice that the ResNet-50 model mistakes are grouped consistently around particular handshapes (compared to the benchmark's model which makes mistakes with a larger variety of handshapes). This is likely because some handshapes, particularly handshapes made with a near closed fist, look relatively similar to one another (see image below).

Considering that this group of handshapes are very similar to one another, it is quite achievement for the ResNet-50 model to be as accurate as it was.

1.5.2 Reflection

The goal of this project was to achieve better ASL handshape recognition than the model outlined in the *Spelling It Out* paper. The same data from the paper of cropped RGB images from five non-native signers were used to train this project's models. The data was first explored and it was determined that the images should be scaled to a consistent square size of 160 pixels by 160 pixels to make it easier to train the model using CNNs. The data were then split into training, validaction, and testing sets to be used to train and test our models.

The first attempt at a predictive model was a convolutional neural network (CNN) built from scratch using three convolution layers with 16, 32, and 64 filters. The model finished with a dense layer using the softmax function of 24 outputs for each handshape to be predicted. This model's best performance was an overall 43% accuracy.

Further attempts were then made using transfer learning. The three model architectures used were VGG16, VGG19, and ResNet-50. For each model architecture, bottle neck features were produced by feeding the images through the model. Then the bottleneck features were passed to a developed CNN to predict the handshape. This CNN was adjusted and fine tuned for each model as well other hyperparameters like batch size and number of epochs. The best performance (overall accuracy) for the VGG16, VGG19, and ResNet-50 models were 66%, 61%, and 70% respectively.

Comparing the best model (ResNet-50) with the benchmark model, it can be observed from each model's confusion matrix that the ResNet-50 model does a better at identifying handshapes over the benchmark model. It particularly does significantly better at similar looking handshapes than the benchmark model. Although the ResNet-50 model doesn't outperform the benchmark on every handshape, it does significantly better overall in prediction.

1.5.3 Improvement

Overall I think there is a lot that could be improved or at the very least could be followed up with future projects. The first area of improvement would be using data that are less limited; the data used included only five subjects all of whom were non-native signers. Part of the reason the why this dataset was used was because there were very few robust/large datasets of ASL handshapes to chose from. There were some datasets that were of ideal quality (comparable to MNIST but for ASL handshapes) but lacked a significant number of data points. There were other datasets, particularly video datasets, but I lack the technical knowledge (computer vision techniques) to extract handshapes from the data.

Having a larger, and varied dataset would likely create a more robust model. I suspect providing a more realistic dataset of native signers to the final model would show the model would not perform as well compared to when the testing set was used. I think providing more data (perhaps by using computer vision techniques) would improve the model's robustness.

The model had difficulty with similar looking handshapes and I think this could be improved by incorporating context with natural language processing (NLP). Similar to how certain phonemes of human speech can be very difficult to identify in isolation, I suspect the similar looking handshapes will always be difficult in identifying in a realistic context of poor resolution or varying angles. If NLP techniques could be implemented to detecting handshapes in a realistic scenario, especially in the use of fingerspelling where words are spelled letter-by-letter, the overall recognition of similar looking hanshapes (letters) could likely be improved.

This project looked at detecting handshapes of associated English letters but could be expanded to full words in ASL if recognition was improved. This is because the handshapes in this project not only correspond to English letters but are essentially characters (possibly comparable to the characters of written Chinese) which are used in full ASL signs which represent full words/concepts. There is much to be improved from this project but this has hopefully become a start of small foothold in the climb of American Sign Language translation.

2 References

[1]: Pugeault, N., and Bowden, R. (2011). Spelling It Out: Real-Time ASL Fingerspelling Recognition In Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, jointly with ICCV'2011.

[2]: Pugeault, Nicolas. "Nico" ASL FingerSpelling Dataset from the University of Surrey's Center for Vision, Speech and Signal Processing, empslocal.ex.ac.uk/people/staff/np331/index.php?section=FingerSpellingDataset.