Landon Buell

Michael Hughes

ENGR 6970

Milestone 02

Artificial Intelligence for Accessibility: Teaching a Neural Network the ASL Alphabet

1. **Abstract**

< TO DO >

1. **Introduction**

The recent advancements of *Large Language Models* (LLM’s) have made the tools and powers of Artificial Intelligence (AI) far more accessible to the public than ever before. However even with text-based chatbots such as the GPT model series, verbal, and auditory AI's such as Apple’s *Siri* and Amazon's *Echo* remain largely unavailable for use in the deaf and hard-of-hearing (DHH) community *<Citation Needed>*. As an early effort to improve the user experience of those non-text/audio-only AI’s for DHH persons, we consider the idea of augmenting such models with the ability to recognize visual/non-audio user input.

In North America, American Sign Language (ASL) is the primary form of communication for most members of the DHH community. It is a complete natural language with many properties of spoken English, but a different grammatical structure. ASL is a visually expressed language where a user conveys information and ideas via a combination of hand and facial movements *<Citation Needed>*. For this study, we produced a proof-of-concept model that teaches two different neural network architectures to categorize and detect characters from the ASL alphabet. The dataset used in this process contains 87,000 RGB images which evenly represent 29 classes. The first 26 are the characters from the Latin Alphabet, and the last three represent spaces, backspaces, and an empty image.

< FINISH THIS >

1. **Methodology**
   1. **Classification Methodology**

A classifier is a type of neural network that solves the problem of associating an entire sample with a particular output class. In the case of image classification, this means that the entire contents of a single input image represent a single category. For example, in our ASL dataset, a signed character may only take up a small subset of the picture’s full body, but we still say that the whole image represents the selected character *<Citation Needed>*. Note that this method is not practical if a single image can be considered to contain multiple classes simultaneously. We design a classification neural network by having the output layer contain as many neurons as the dataset contains classes. When an input is passed through the model, we normalize the output layer such that L1-norm is equal to 1. This way the value contained within the *n-th* neuron (also called the *activation* of that neuron) is the probability of that input sample belonging to the *n-th* category.

< INTRODUCE / MOTIVATE ARCHITECTURE HERE>

* 1. **Segmentation Methodology**

An image segmentation model is a type of neural network that solves the problem of first detecting that an instance of a category exists within an image, and then further isolating the subset of pixels correspond to that class. Image segmentation is like a refined object-detection algorithm, with the added task of showing what pixels and shapes make up that instance of that class *<Citation Needed>*. Image segmentation is valuable because it handles the case where in input image may contain one or more instances of the same class, or multiple different classes simultaneously. *<What does the output of an image segmentation model look like?>*

< INTRODUCE / MOTIVATE ARCHITECTURE HERE>

* 1. **Preprocessing Strategies**

Preprocessing input samples is the act of refining, organizing, and cleaning each sample in a dataset before it is presented to a machine learning model for processing. This is critical because it allows us to tailor our dataset by standardizing the scale of all samples and removing anomalous or extreme-outlying datapoints. This prevents a model from accidentally overfitting or learning from any of the quirks that may arise out of raw or untidy data.

For preprocessing out ASL dataset, we subject each input image/batch to the following preprocessing procedure, and motivate that step.

* + 1. Image Cropping – Crop the 8 outermost pixels from the top, bottom, left, and right sides of each sample. This brings the input size from (3 x 200 x 200) down to (3 x 184 x 184). From manually exploring the dataset, we can see that each image is in general centered within the frame, therefore removing an 8-pixel border removes no vital information while also greatly reducing the input dimensionality of the model. This also has the added bonus of allowing for fewer parameters withing the rest of the neural network.

*< Show Before & After Image>*

* + 1. Image Normalization – Scale each channel of the input image such that it has a mean of 0, and a variance of 1. This reduces any potential skewness that may arise within a given sample.

*< Show Before & After Image>*

* 1. **Augmentation Strategies**

To ensure a robust model that is capable of differentiating between classes reliably and consistently and capable of handling large variations of data within a class, it is common practice to artificially increase the size of the dataset. This is done by making copies of each training sample that are slightly modified with some transformation while keeping the original class label. This process is called *Dataset Augmentation* *<Citation Needed>*. Augmentation is possible and so frequently used in image processing and recognition tasks because the subject in each image sample is often highly invariant to perturbations. For example - a cat is still a cat regardless of where it appears in an image, a Rocketship is still a Rocketship regardless of which direction in faces in an image and a television is still a football player is still a football player regardless of their team’s colors.

Classifier and segmentation models can benefit from this augmentation strategies because it can greatly increase the size of the training datasets while also enabling the models to account for much larger in-class variances. Additionally, image-processing models often contain a high number of parameters which can require a massive amount of training examples before is has sufficiently converged. <FINISH THIS>

* 1. **Performance Metrics**

There are multiple ways to score the performance of a classifier model for image recognition and segmentation tasks. Below are the metrics that we choose to use to evaluate the behavior of our models along with their definition and a description of how they work. <TO DO>

* + - 1. Accuracy Score – A basic and often uninformative metric which compares the number of correct classifications to the total number of testing samples processed. Accuracy is not commonly used because it provides little or misleading information as to what is happening on a class-by-class basis.
      2. Precision Score - <TO DO>
      3. Recall Score - <TO DO>
      4. F1 Score – Harmonic mean of precision and recall, bounded between 0 and 1. A higher score means that a model is in general performing correct classifications.
      5. Confusion Matrix – A square matrix with the number of rows and columns equal to the number of classes. The number in row “a” and column “b” means the number of samples in class “a” that were predicted to belong in class “b”. A Confusion matrix with a strong main diagonal (i.e. a = b) means that the model was consistently making correct classifications.
      6. Rate of Convergence – Tracks the rate at which the objective function of a model decreases as training progresses. A high rate of convergence means that the model quickly optimizes and finds a reasonable set of parameters that can map inputs to output classes.

1. **Experimental Results**

< TO DO >

1. **Conclusion**

< TO DO >

1. **Works Cited**

< TO DO >