Convolutional Neural Networks for ASL Image Classification

P37: Matthew Martin, Nitesh Mishra, Aneesh Sreedhara, Sumit Singh
Department of Computer Science, North Carolina State University
Raleigh, NC 27695
{mkmarti5, nmishra4, asreedh, ssingh57}@ncsu.edu

1 Background

1.1 Problem

Hearing loss is an issue that affects a significant portion of the American population. American Sign Language (ASL) - a visual language - is a common communicative tool for people who are deaf or hard of hearing. However, this is not understood by the majority of the people who are non-signers. We aim to bridge this barrier by enabling efficient conversation between them by creating and training a classifier to properly identify the sign language symbol that is present in an image input. The potential of such a tool can be seen in a variety of use-cases, such as the real-time translation of ASL to text.

The major focus of our research is regarding the use of Convolutional Neural Networks (CNNs) to solve this image classification problem. Through applying these types of networks to our ASL image classification task, we hope to gain a better understanding of how the design of a neural network can impact its accuracy of predictions. In our experiments, we vary:

1. The number of convolutional layers in our network.
2. The number of fully connected layers in our network.
3. The pooling strategy applied in the convolution portion of our network.

We hypothesize that each of these aspects can and will have a significant impact on the outcome accuracy of each network. Our goal is to deduce an optimal structure for a Convolutional Neural Network for ASL image classification.

1.2 Literature Survey

When looking into solving our problem, we first surveyed popular existing strategies for image classification. A basic solution for image classification revolves around the idea of identifying features within an image that can be used to estimate what is contained within the image. In a traditional classification structure such as a decision tree or k-nearest-neighbors classifier, these features would need to be manually identified throughout training data "to reduce the amount of visual feature data to describe the entire image at the highest possible level," [1]. This can be a time-consuming and cumbersome task that is not necessarily practical for such a large dataset as the one that we have selected.

Our continued research led us to a more modern solution in Convolutional Neural Networks. According to Anne Bonner, "A CNN convolves learned features with input data and uses 2D convolutional layers. This means that this type of network is ideal for processing 2D images," [2]. She goes on to say that these types of neural networks automate the process of feature extraction, meaning that a user

does not have to manually define and extract features from a series of images, one of the major pain points of using other image classification techniques as mentioned previously. This ease of automated feature extraction led us to the conclusion that CNNs would be an appropriate classification strategy for us to further explore in our studies of the ASL image dataset.

2 Method

Images depicting symbols within the American Sign Language will be utilized to train, validate, and test various convolutional neural networks with varying layer depths and activation functions to verify the impact of such variations on the performance of a model, primarily evaluated by the model's accuracy. Convolutional neural networks are an industry standard for image recognition tasks owing to their impressive performance with image input, thus making them the ideal class of model for the task at hand.

2.1 Data Selection

Data is collected from the ASL Alphabet data set on Kaggle (`https://www.kaggle.com/datasets/grassknoted/asl-alphabet`). This set contains two sets of data for training and testing models. The given training set contains 87000 square images with a side length of 200 pixels, and can be categorized into 29 classes, one class for each letter A through Z, and classes for *SPACE*, *DELETE*, and *NOTHING*. The given testing set contains 29 images.

2.2 Preprocessing and Transformation

We take the given training set and split it into three sets for training, validation, and testing. Of the original training set, 60 percent is used for training, 20 percent for validation, and 20 percent for testing. Inputs are standardized for compatibility with convolutional neural networks.

2.3 Model Creation

Various models of convolutional neural networks will be created, each composed of multiple, sequential layers. Each model may be divided into three sections of layers: a convolution-pooling section, a flatten section, and a fully connected section; such an architecture is common among models of this type. Variations are made in the depth of models' convolution-pooling section, the depth of models' fully connected section, and layers' activation functions. The aforementioned sections of each model are described in greater detail below.

Figure 1 shows the basic structure of a convolutional neural network, with each of the key components highlighted in a different color [3].

Convolution-Pooling Section

The convolution-pooling section is composed of multiple sets of convolution and pooling layers that perform feature extraction and dimensionality reduction. Importantly, these operations preserve spatial relationships within images, which will be critical in later sections. Average pooling and max pooling are two common pooling methods that summarize the average presence of a feature and the most activated presence of a feature respectively.

Flatten Section

The flatten section employs a flatten layer to flatten the output of the convolution-pooling section for compatibility with the fully connected section. It will take the 2-dimensional input given to it and transform it by enumerating the values of the 2D input into a single, 1-dimensional vector.

Fully Connected Section

The fully connected section utilizes a number of fully connected layers to form a multi-layer perceptron, the output of which is used for classification.

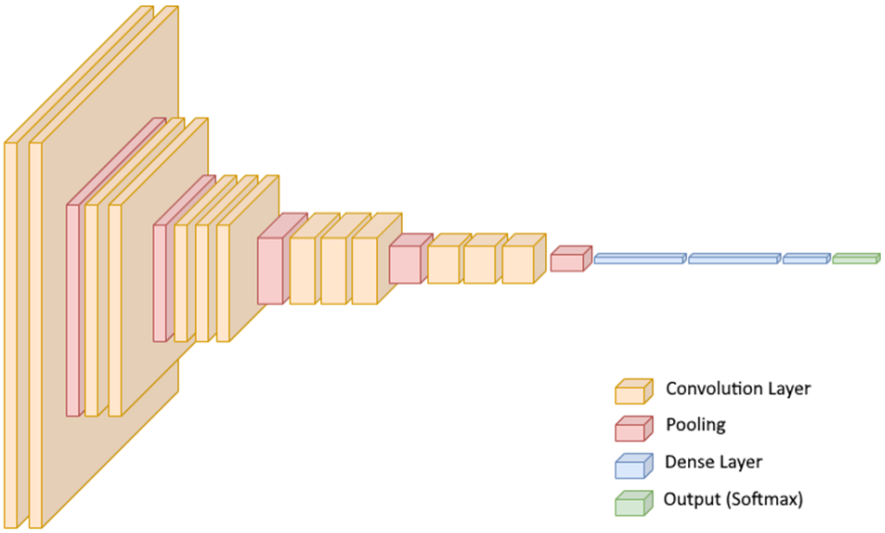

Figure 1: The basic structure of a convolutional neural network for image classification. [3]

2.4 Model Evaluation

Models are evaluated primarily by their accuracy. The decision to prioritize accuracy is based on the rationale that the primary goal of these ASL image classification models is to correctly classify inputs, and that there is no major cost associated with any certain type of incorrect classification, seeing as how any incorrect classifications would likely only result in slight confusion to observers viewing the classifications.

3 Plan & Experiment

3.1 Datasets

Our project uses the ASL Alphabet dataset present on Kaggle. The dataset contains images for the letters A through Z, space, delete, and blank images separated in different folder. There are 3000 sample images for each class, totaling to 87000 images overall.

3.2 Hypotheses

We constructed three hypotheses to test in our experiments:

1. Changing the number of convolutional layers in the neural network architecture has a significant impact on the accuracy of our system.
2. Using the max pooling strategy in our neural network has a significant impact on the output accuracy of our system.
3. Changing the depth of the fully connected section of the neural network has a significant impact on the output accuracy of our system.

3.3 Experimental Design

In order to get answer to each hypothesis we have designed certain experiments. For each of the experiments, any aspects of the network not explicitly mentioned to be changed remain constant in structure for each test.

Hypothesis 1:
We analyzed the results obtained by the model architecture by changing the number of convolutional layers in our neural network architecture. We experimented using 1, 2, 3, and 4 convolutional layers in the network structure, and collected output metrics of accuracy for each model on the testing dataset.

Hypothesis 2:
We compared the results obtained by the model architecture by using either the max pooling or average pooling strategy within our neural network architecture. We collected the output metric of accuracy on the testing dataset for each network that we trained.

Hypothesis 3:
We analyzed the results obtained by the model architecture by changing the number of densely connected layers in our neural network architecture. We experimented using 1, 2, 3, and 5, and 10 densely connected layers in the network structure, and collected output metrics of accuracy for each model on the testing dataset.

4 Results

We had varying results for each of our experiments. Each experiment is discussed below in isolation to each of the others, as the different networks can only be accurately compared with others in their same category. This is because each experiment isolates a single element of the network structure to identify its impact on its overall performance.

However, we did try to use a unifying baseline structure for our networks throughout all tests. Except for the experiments in which these specific values varied, our networks used three convolutional layers with a ReLU activation function and a max pooling layer in between each convolutional layer. This portion of the network was followed by flattening and two dense layers using ReLU and Softmax activation function for classification.

4.1 Hypothesis 1: Changing Convolutional Layers

Figure 2 shows the confusion matrices generated through testing networks with varying numbers of convolutional layers on an external testing data set that was not used at all during training or validation. Overall, all of the models had very high accuracy at 96% or higher. The accuracy did continue to rise as we increased the number of convolutional layers. Overall, we concluded that our experiment did show that the number of convolutional layers has a significant impact on the accuracy of the model, but this impact was not as pronounced as expected and had diminishing returns. In addition, we had originally expected that accuracy would hit a peak and then fall after reaching a certain number of layers. This is because our input becomes so condensed in the summarizing process of convolution and pooling that we assumed that we would begin to lose information along the way. This still may be the case, but would have to be tested in the future with higher numbers of convolution and pooling layers.

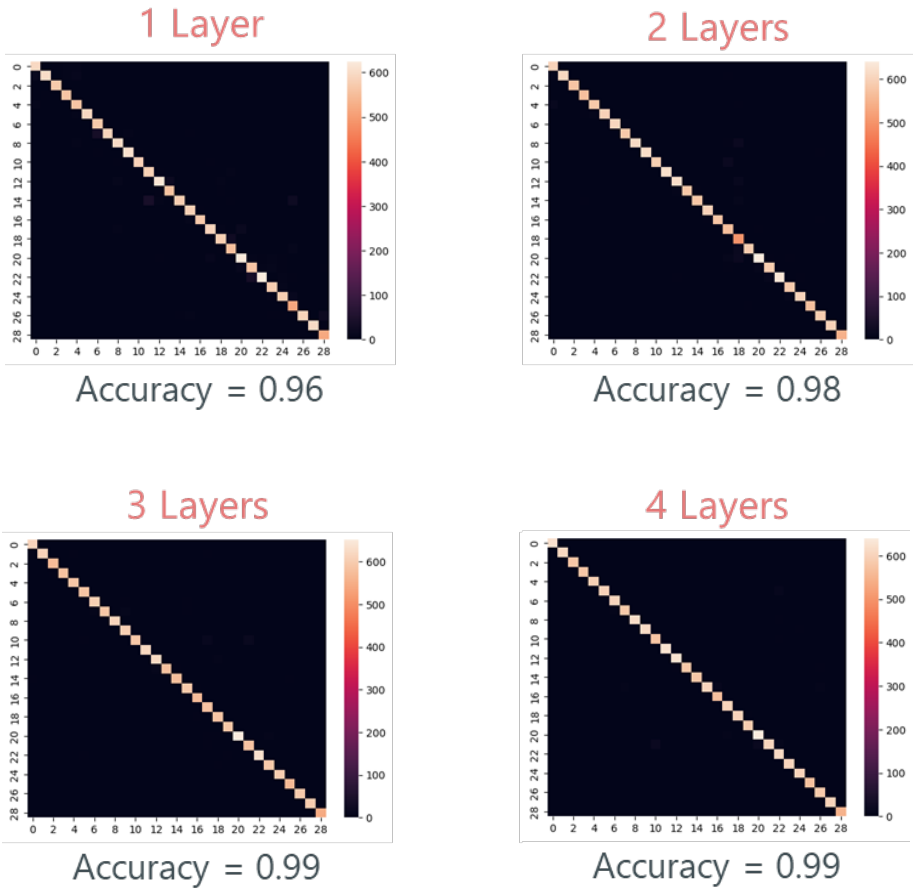

Figure 2: The confusion matrices generated during testing of models with varying numbers of convolutional layers.

4.2 Hypothesis 2: Changing Pooling Strategy

Figure 3 shows the confusion matrices generated through testing networks with different pooling methods sandwiched between the multiple convolutional layers on an external testing data set that was not used at all during training or validation. Overall, we had two separate models for this task, one based on max pooling and the other on average pooling. Both models showed excellent output accuracy well over 95%. However, max pooling showed better accuracy of 98% over average pooling with 96%. This result can be interpreted as max pooling performing slightly better over average pooling. In theory, max pooling selects the brighter pixels and is useful when the image is dark. However, average pooling smooths out the image and hence the sharp features may not be identified. This may be the case as to why max pooling performed better given the solid theoretical background support, but would have to be tested in the future with a dataset not involving images with a darker background which would make an even playground with average pooling. Also, it would be interesting to work with a combination of different pooling strategies in between the convolutional layers.

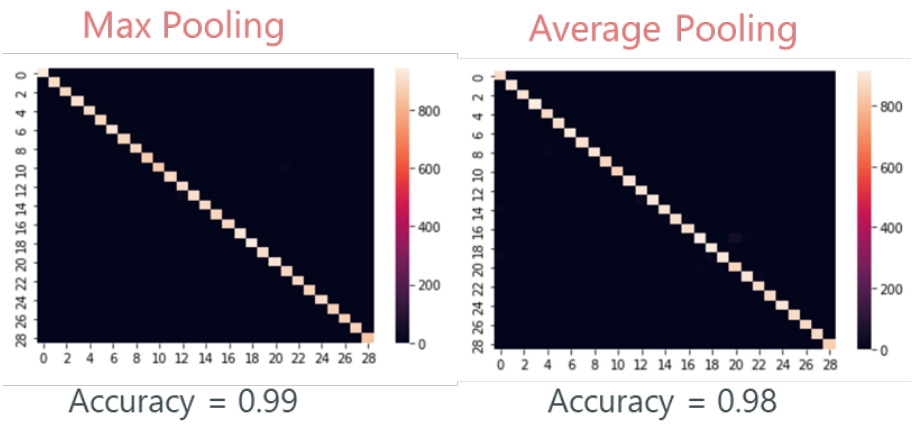

Figure 3: The confusion matrices generated during testing of models with varying pooling strategy.

4.3 Hypothesis 3: Changing Fully Connected Layers

Figure 4 shows the confusion matrices generated through testing networks with varying numbers of fully connected layers on an external testing data set that was not used at all during training or validation. Accuracy remained consistent across the models, slight variations aside. We originally expected to see a peak then subsequent decline in accuracy as the number of fully connected layers met then surpassed an appropriate amount resulting in over-fitting. In our case, merely one layer was appropriate, but contrary to our expectations, accuracy remained consistent throughout. This suggests that the number of fully connected layers, past an appropriate amount, does not significantly impact model performance by our metrics. It should be noted, though, that the model with one fully connected layer is still "better" than the model with ten fully connected layers, following from Occam's razor. We suspect that the continued addition of layers would eventually lead to over-fitting and a decrease in performance, but this is left for future study with the resources needed to create and test models with significantly more layers.

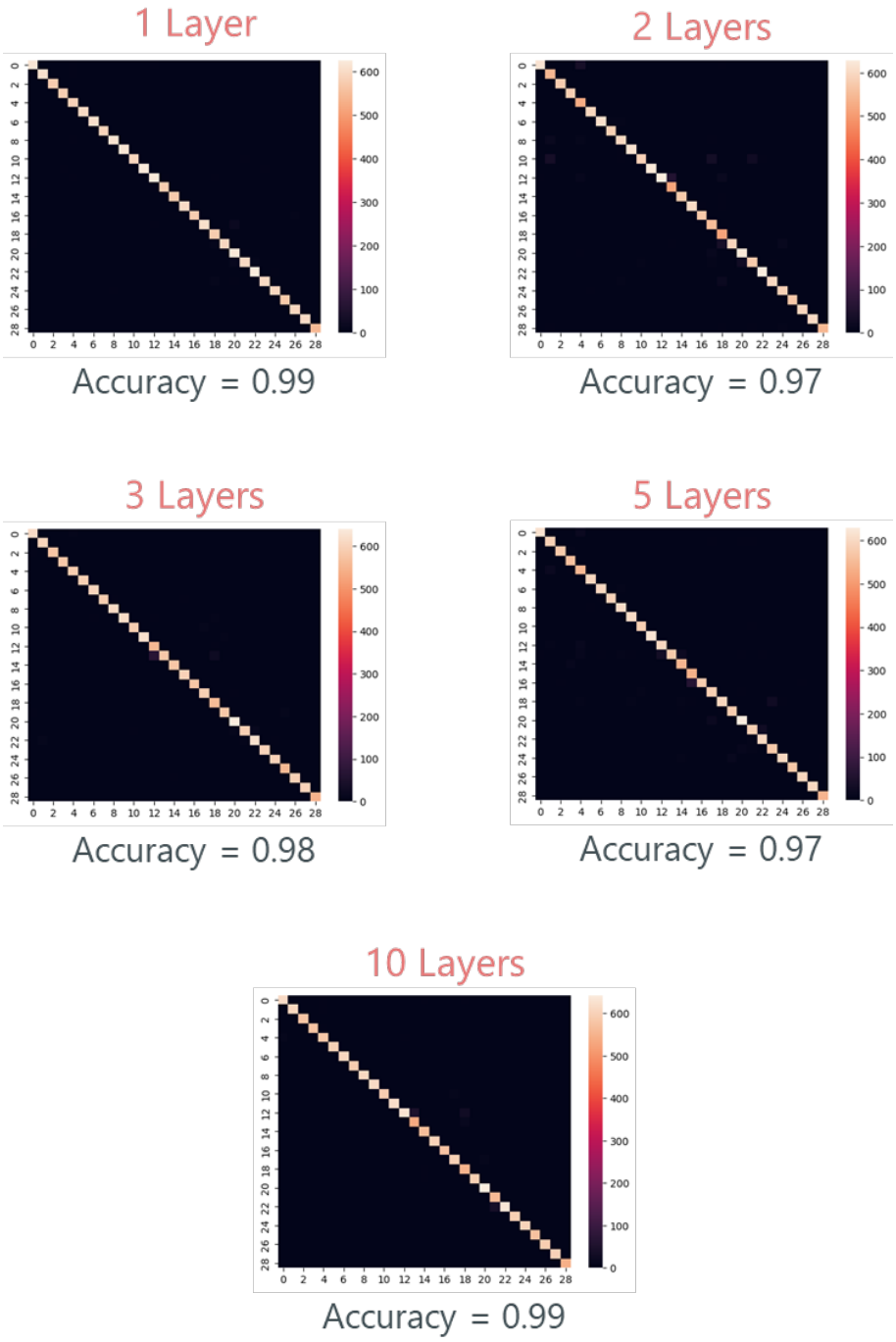

Figure 4: The confusion matrices generated during testing of models with varying numbers of dense layers.

5 Conclusions

There were several things that our team learned throughout the process of completing this project. Firstly, the task of image classification is a surprisingly approachable task for an individual that has access to a dataset that is properly labeled. There are plenty of tools available, including machine

learning libraries like TensorFlow and computing resources such as Google Colab that empower individuals to complete these tasks. Google Colab provided our team with the appropriate amount of computing power to be able to complete this task without having to personally invest in powerful machines to take on the task.

Secondly, we had interesting findings on the applications of neural networks. It seems that one of the key tasks in creating a neural network that is well designed for the task includes choosing a proper network structure. In addition, the choice of network structure can have significant impacts on the amount of time required to train the network. Some of the networks that we trained took multiple times as long to complete the training process as others, likely due to the wildly varying number of parameters to assign values to in each model.

Thirdly, we were able to observe the theory of Occam's razor in action. Past a certain point, the addition of complexity only resulted in diminishing returns, if any at all. This reaffirmed the idea of not creating models that are any more complex than they have to be. If you are adding complexity and see model performance begin to plateau, then you should reconsider continuing to add that complexity.

Finally, we found that neural networks take a lot of input training data to be successful. Our team used a dataset of 87000 images to complete our task, with a total file size footprint of 1.11 GB. It is clear that having a large dataset was helpful to our model performing well, as more data allowed for a larger pool of records for our model to learn from.

Access our Code

All of our code for this project was completed using Python, Tensorflow, and Jupyter Notebooks. We executed our Jupyter Notebooks using Google Colab to allow for easy access to free, powerful machines. You can access our code, as well as saved model files that were generated, at the following GitHub repository: https://github.ncsu.edu/mkmarti5/ALDA2022_ASLClassification_P37.

References

[1] Scherer, Rafał. Computer Vision Methods for Fast Image Classification and Retrieval. 1st Edition, Springer International Publishing, 2020, SpringerLink, https://link.springer.com/content/pdf/10.1007/978-3-030-12195-2.pdf.

[2] Bonner, Anne. "The Complete Beginner's Guide to Deep Learning: Convolutional Neural Networks." Medium, Towards Data Science, 1 June 2019, https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb.

[3] Leung, Kenneth. "How to Easily Draw Neural Network Architecture Diagrams." Medium, Towards Data Science, 13 Sept. 2022, https://towardsdatascience.com/how-to-easily-draw-neural-network-architecture-diagrams-a6b6138ed875.