Comparison of Models Trained on American Sign Language (ASL) Character Recognition

Ben Lubas, Derrick Kim, Sunny Huang

Northeastern University, College of Computer and Information Sciences

December 14, 2022

Abstract

American Sign Language is a system of hand signals that allows people to communicate by making signs with their hands. ASL has signs for each letter of the alphabet. The goal of this project is to train and optimize a model to determine the alphabetic letter corresponding to the given hand signal. A convolutional neural network (CNN), a convolutional neural network with a dropout layer, and a feed forward neural network (FFNN) were trained and compared with each other to see which model performs the best at this task. The impact of cross-validation strategies were also evaluated with the networks being trained on both a train-test split approach as well as k-folds cross-validation. We found that the convolutional neural network performs best, especially with k-folds cross-validation.

1 Introduction

Image classification is a very common supervised learning problem that has practical applications in a variety of fields. Our project aims to determine which type of network, CNN or FFNN, is best suited for recognizing ASL characters, which also gives insight as to which networks perform best in other image classification problems.

1.1 Problem Statement

The goal of this project is to determine which networks can best tackle this problem and achieve high accuracy in recognizing images of ASL characters. Each network will take a 28x28 grayscale image p as input, and output a vector l, where each index i corresponds to a letter, and the value l_i represents the likelihood of the input being the corresponding letter.

1.2 Project Relevance

This project would be useful if someone who does not know ASL needs to translate a series of ASL hand signals (specifically letters) into English letters. Since the models do not recognize words in ASL, this project is more useful as a learning experience, and a base for a more capable machine learning model that can recognize entire ASL words. Such a model would be much more complex. It would need time series data because ASL signs depend not only on the shape of the hands. Additionally, the images would have to be framed wider because signs depend on hand movements and position relative to other people.

2 Technical Approach

The objective of this project is to train multiple models to recognize ASL hand signals, and compare these models against each other to determine the best model. Each model was trained first using the same hyperparameters as the other models, and the testing accuracies were compared. Each model was then

tuned and optimized, and the accuracies were compared once again.

2.1 Convolutional Neural Network

Our Convolutional Neural Networks were broken down into one with a dropout layer and one without. The basic architectures of the models were the same. We had 2 convolutional layers where one had 64 filters and the other had 32 filters. Both convolutional layers had a kernel size of 3, stride of 1, and padding of 0. Each convolutional layer also used a rectified-linear unit (ReLU) activation function. Following the convolutional layers we had one max pooling layer which downsampled the image by a factor of 2. From there the image was fed into a fully-connected layer of 128 neurons outputting 26 values, one for each class. Our fully-connected layer also used a ReLU activation function.

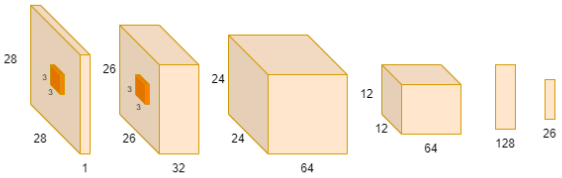


Figure 1: Convolutional neural network diagram

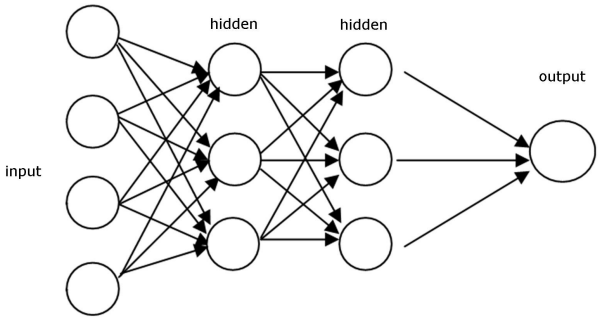
We decided to also implement a Convolutional Neural Network with a dropout layer because of the overfitting we experienced training the initial CNN. We included this network in our experiments to see if it was able to tackle the problem of overfitting in our dataset. Our dropout layer randomly selects a portion of the nodes to drop out of the network. The outputs of the dropped nodes would no longer be the inputs of the next layer, reducing node interdependence and ideally preventing overfitting of the training data and improving the test accuracy. In practice, the dropout layer did not greatly affect the test accuracy of the CNN. In fact, the CNN's performance without the dropout layer was slightly better than the CNN with the dropout layer when trained with both a train-test split and with cross validation.

2.2 K-Folds Cross Validation

In order to reduce bias in our dataset when training, we decided to use k-folds cross-validation. Specifically we used folds of size 4, partitioning the dataset into 4 equal sections. For each iteration up to k iterations, one section was chosen as the test dataset while the other sections were used for training the model. The overall accuracy was then determined by averaging the performance of all 4 models. At each fold, the model is reset, so there is no influence from previous iterations. Our CNN, CNN with a dropout layer, and FFNN models were all trained using the cross validation method, and the test accuracies were then compared to each other and to the results when we trained with a train-test split.

2.3 Feed Forward Neural Network

For the sake of comparison, we decided to incorporate a Feed Forward Neural Network in our experiment. We expected this network to perform poorly, or at least worse than the two Convolutional Networks that we trained. The model was trained using the same methods that the CNN was trained with, and also tested using the same method. The test accuracy of the FFNN was then compared to that of the CNN.



[2] Figure 2: Simple feed forward network diagram

As seen in Figure 2, all the nodes in each layer are connected to all nodes in the next layer. This sort of connection means that each node is influenced by every pixel in the input layer, rather than focusing on local areas of the input image. The diagram above is a simplified network with fewer nodes. The FFNN used in this project has 784 nodes in its input layer

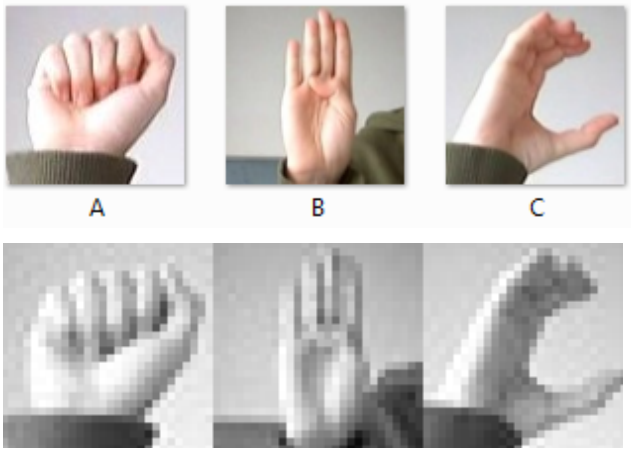
representing the 28x28 pixel image, then two hidden layers, each with 128 nodes, and finally an output layer with 25 different nodes. Each node represents a character, and the value output is the likelihood of the input being the corresponding character.

3 Experimental Results

The general process used in this project consists of training and tuning a model, then extracting its test accuracy and comparing it against the other models. We found that out of the three classification models, the CNN without a dropout layer, trained with cross validation performed the best on the given test data.

3.1 Dataset

The datasets for hand signals are taken from the Sign Language MNIST Library, which was provided on Kaggle. The given train and test datasets are both csv files which contain vectors of length 785, where the first entry of the vector is a label 0-25, and the other 784 entries are the pixel values of the 28x28 image. The images were already preprocessed for us, being grayscale and thus having only 1 channel, as shown in Figure 3. For input into our networks, the images were either reshaped into a 28x28 image for input into our Convolutional Neural Networks or kept as a vector input for our Feed-Forward Neural Networks. The dataset includes 25 total classes, each representing an alphabetic letter as well as a class representing “no signal”. The classes for J and Z were omitted because their ASL translations involve movement in addition to the usual hand sign.



[1] Figure 3: Example ASL images processed down into 28x28 grayscale images

3.2 Training

Two different methods were used to train the CNN, CNN with a dropout layer, and the FFNN. The first method used was a train-test split. Using a mini-batch size of 16 that was randomly generated by a data loader, each model was trained over five epochs with 100 iterations in each epoch. After the training was completed, the model was run on a test batch, which was used to determine the test accuracy of the model. During the training process for all three models, cross entropy loss was used for the loss function, and Adam was used as the optimizer. We found that the loss for the FFNN was significantly higher than that of the other models, as shown in Figure 4. The loss for the two CNN models was basically zero.

We also implemented a k-folds cross-validation to decrease overfitting and hopefully increase test accuracy. We used folds of four, as explained in 2.2. The average accuracies of the CNN and CNN with dropout were, once again, significantly higher than that of the FFNN. We used the same loss function (cross entropy loss) and optimizer (Adam) when training with cross validation. The results are shown in Figure 5.

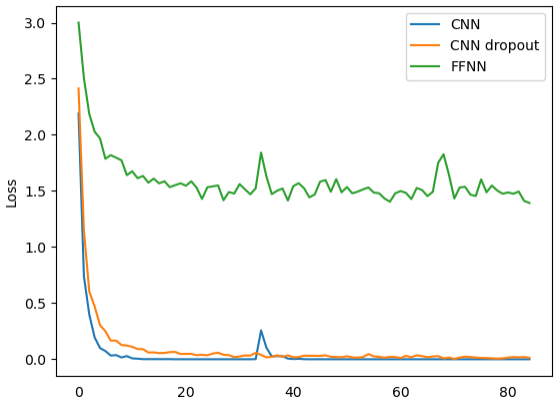


Figure 4: Loss plot for each of the models over the training process, trained with classic train-test split.

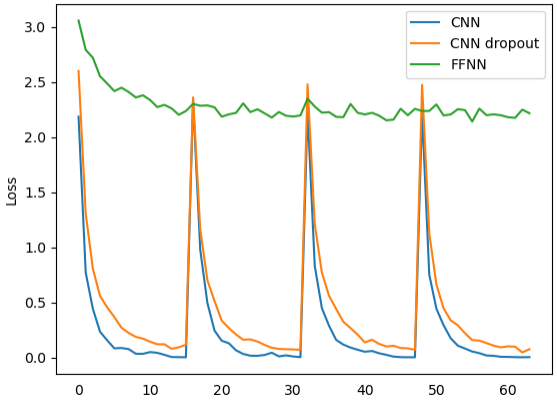


Figure 5: Loss plot for models trained with cross validation.

As seen above, the shape of the loss plots are very different when training with cross validation compared to a train-test split. At the beginning of each fold, the loss is extremely high, but falls quickly, since each fold is only trained for a single epoch.

3.3 Testing

The testing methods were kept the same for each model to limit any variations in accuracy due to confounding variables. Each model was tested using data passed in from a data loader. However, unlike the training process, the data loader did not randomize the data being passed into the testing to limit any variance in the test accuracies that could be caused by randomness. The trends we found were close to what we originally expected.

When using a train-test split, the accuracy of the CNN model was 0.897. The accuracy was about the same at 0.869 when the dropout layer was added. The accuracy of the FFNN was significantly lower at 0.413. The test accuracies for the CNN models were not as high as we expected. Since the training loss was nearly zero, we expected the models to perform almost perfectly. This means that the CNN were most likely overfitting, which was solved when we implemented cross validation.

When using cross-validation, we found the trend between the models to be the same, with CNN performing better than the FFNN. However, the overall accuracy of all three models rose when using cross-validation, indicating that the model was no longer overfitting. The CNN and CNN with a dropout layer had test accuracies of 0.998 and 0.973, respectively, while the FFNN had an accuracy of 0.612. CNN clearly performed better than

4 Conclusions

This paper presented the results of training multiple types of neural networks to recognize alphabet letters based on the hand signals. A convolutional neural network and a feed forward neural network were trained and compared against each other. We found that a standard CNN, described in 2.1, performed significantly better than a FFNN, described in 2.3. The accuracies of each model trained with each method are noted in 3.3. The results show that the models, when trained with a train-test split, were clearly overfitting, as the training losses were significantly better than the test accuracies. This disparity was solved with k-fold cross validation, described in 2.2. The models, once trained with cross validation, had test accuracies that better matched their respective training losses. Overall, this project succeeded at building a model to recognize ASL characters, as well as comparing the performance of different models.

Future work for this project should include enhancing the current CNN model to include the missing letters J and Z, and also to include words, not just letters. This would be a challenging task, as it would require many more output classes. It would also require a more complex input that includes time series data. A possible way to implement this would be to separate and downscale videos into a few frames of 28x28 pixel images, which would represent how the hand moves over time. The model might then be separated to first categorize the input as a word or a character, then as a more specific type of word (noun, verb, etc.), then finally converted to an output that includes the probabilities of the ten most likely words/letters. These enhancements are outside the scope of this project, but would make the model much more powerful and practically relevant. We would be able to use it to translate ASL in real time, as well as a variety of other applications.

5 Contributions

The work for this project was divided evenly among the contributors. We all worked together for the research, programming, and report portions of the project, and there was no clear task assignment for each member.

References

[1] Tecperson. “Sign Language MNIST.” *Kaggle*, 20 Oct. 2017, https://www.kaggle.com/datasets/datamunge/sign-language-mnist.

[2] “Feed Forward and Feedback Networks.” *Packt Subscription*, https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788397872/1/ch01lvl1sec21/feed-forward-and-feedback-networks.

[3] Ng, Ritchie. “Feedforward Neural Network with Pytorch.” *Feedforward Neural Networks (FNN)*, Deep Learning Wizard, https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_feedforward_neuralnetwork/.