CS 412 Final Project

Group – 13

Berkay Savcı – 25381, Ece Kurnaz – 26727

Mert Ertörer – 23656, Yusuf Araz – 25479

In this project, we are required to train machine learning network to determine a sign language letter in images. As training data, we have 29000 labeled images which are 200x200 pixels and we will use our trained model on images with same sizes. Instead of training all the model from scratch 2e have decided to use Transfer Learning. Transfer Learning enables us to transfer knowledge acquired for different domain by taking a network trained for that domain with larger datasets and classes, then adapting to our sign language domain by editing last steps of the model. When implementing Transfer Learning have used ResnetV50 which is a CNN model trained over 50,000 images as our base model. The advantage of using this model is pretrained weights, therefore our model makes small changes on weights to get the best results. However, after defining base model we need to integrate the model to our problem domain by adding fully connected layers at the end of the model. Therefore, we have tried to add pooling layer, dropout layer and hidden network layers with different hyperparameters to enhance our accuracy. In the rest of the report, we will describe the various implementations we have made and compare their effects on accuracy. Different variances of our models and their performance in terms of validation accuracy is also given provided Colab document.

Firstly, we have decreased our sample size to 100 images per one letter to have faster iterations since training a model with all of the data consumes approximately 3 hours and would not be applicable for hyperparameter tuning. Then we have normalized our data before training. Afterwards we have added a max pooling layer on top of 50 layer of Resnet to keep most important features of the image while reducing dimensionality to increase train efficiency. After pooling layer, we have tried to add 1, 2 and 3 fully connected layers with different number of neurons. Furthermore, we have trained these models with dropouts that have different dropout rates between hidden layers which resulted in decrease in accuracy. However, we have decided to include dropout to our final model since we are trying different models with sampling 10 percent of our train data and want to ensure we don't overfit when training with all data. Lastly, we have added our output prediction layers which have of course 29 layers. Other hyperparameters that we have changed to optimize our models are learning rate, loss function and number of epochs. For determining number of epochs, we have used early stopping which is a method that allows you to specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on a hold-out validation dataset. We have tried learning rates 0.1, 0.01 and 0.001. Large learning rates result in unstable training and tiny rates result in higher time complexity since it improves baby steps in each iteration. If the learning rate is too large, weight updates will over-shoot the min, there will be too much oscillation. If too small learning will take too long but generally, decay in the learning rate over iterations is used commonly. We have found that 0.001 is ideal for our models. We also have used another activation function 'sigmoid', and get better accuracy compared to 'relu' activation function. However, using sigmoid function increases train time dramatically which led us to continue with 'relu' activation function.

In conclusion, we get the best model with using relu activation function, 0.001 learning rate, one hidden layer with 256 neurons, on top of our base model and pooling layer in our trials. Then we have added dropout to our final model to not overfit when training with sample size 1000. The comparison between different models is provided in following tables. Moreover, we have uploaded 2 Colab Repository for final best model and comparison of our previous models which are named accordingly.

COMPARISON TABLES

NUMBER OF LAYERS	ACCURACY WITH DROPOUT	ACCURACY WITHOUT DROPOUT
ZERO HIDDEN LAYER	0.9517	0.9569
ONE HIDDEN LAYER	0.9534	0.9621
TWO HIDDEN LAYER	0.9414	0.9603
THREE HIDDEN LAYER	0.9172	0.9345

LEARNING RATE	0.001	0.01	0.1
ACCURACY WITH ONE HIDDEN LAYER	0.9603	0.8586	0.0448

NEURON NB	ACCURACY IN ONE HIDDEN LAYER	ACCURACY IN TWO HIDDEN LAYER	ACCURACY IN THREE HIDDEN LAYER
UPPER LAYER HAS 512 NEURONS	0.9569	0.9586	0.9259
UPPER LAYER HAS 256 NEURONS	0.9621	0.9603	0.9172

ACTIVATION FUNCTION	ACCURACY
SIGMOID (IN ONE LAYER)	0.9690
RELU (IN ONE LAYER)	0.9603