CSE 574: Programming Assignment 5

Vivek Shankar
Department of Computer Science
University At Buffalo
shankar9@buffalo.edu
&
Megha Choudhary
Department of Engineering & Applied Sciences
University At Buffalo
meghacho@buffalo.edu

Aishwarya Agarwal
Department of Engineering & Applied Sciences
University At Buffalo
aa299@buffalo.edu

May 18, 2022

Abstract

The task of this project is to perform classification using Machine Learning. We will be solving two class problem. The task is to classify what ASL word is performed by an ASL signer based on the acoustic images provided in the data set,using CNN as the classifier.

1 Dataset

The dataset is provided by our current research project, which include 10 ASL words performed by 5 subjects. In this dataset, all images are generated by using the short-time Fourier transform (STFT) to calculate a spectrogram as the feature representation of the reflected near-ultrasound waves. Based on the Doppler effect, sign language gestures, including both hands and arms, will cause phase and frequency changes of the reflected sonic wave. The spectrogram contains information in both frequency and time domains. The spectrogram is also defined as the Power Spectral Density of the function:

$$\text{spectrogram}\{x(t)\}(\tau,\omega) \equiv |X(\tau,\omega)|^2 = \left|\sum_{n=-\infty}^{\infty} x[n]\omega[n-m]e^{-j\omega n}\right|^2$$

where x[n] is input signal, and w[n m] represents the overlapping Kaiser window function with an adjustable shape factor β that improves the resolution and reduces the spectral leakage close to the sidelobes of the signal. The coefficients of the Kaiser window are computed as:

$$\omega[n] = \frac{I_0\left(\beta\sqrt{1-\left(\frac{n-N/2}{N/2}\right)^2}\right)}{I_0(\beta)}, 0 \leq n \leq N$$

This dataset has a training set of 5,000 examples, and a test set of 1,000 examples.

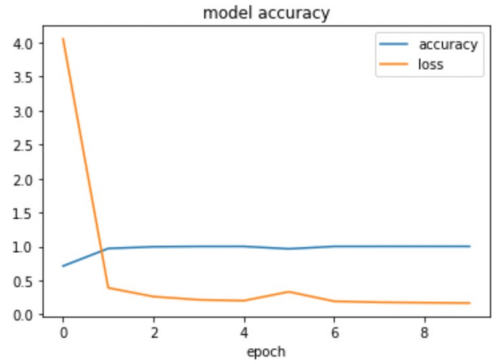
2 Results

2.1 Building a CNN with 2 or 3 hidden layers without regularization methods

Test loss = 1.0069
Test accuracy = 0.875

2.2 L1 Regularization

```
Test loss: 1.2191401720046997
Test accuracy: 0.875
```



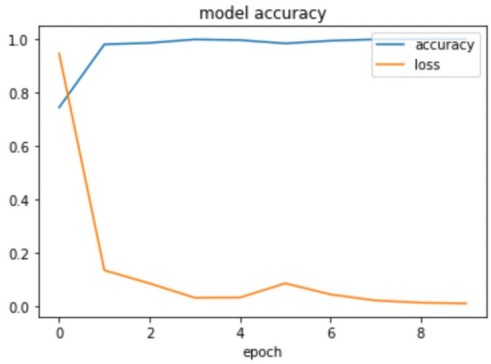
While Training, for L1 regularization we used 10 epochs:

For Epoch 1/10
Loss = 4.0545
Accuracy = 0.7107
...
For Epoch = 10/10
Loss =0.6947
Accuracy = 0.8693

While Testing
Loss =1.2191
Accuracy = 0.875

2.3 L2 Regularization

```
Test loss: 1.1291062831878662
Test accuracy: 0.8125
```



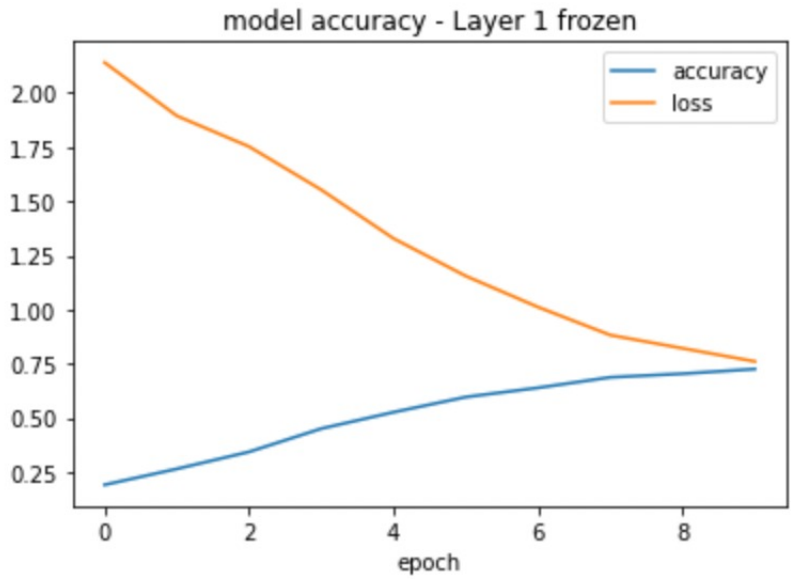
While Training, for L2 regularization we used 10 epochs:
For Epoch 1/10
Loss = 0.9462
Accuracy = 0.7445
...
For Epoch = 10/10
Loss =0.6409
Accuracy = 0.8578

While Testing
Loss = 1.1291
Accuracy = 0.8125

2.4 Resnet

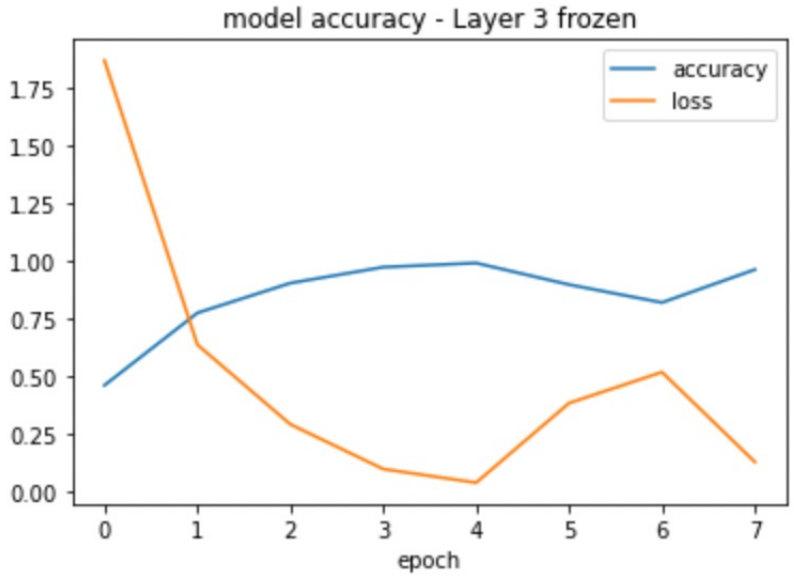
1. Model Accuracy keeping Layer 1 Frozen

```
Test loss: 0.7332940101623535
Test accuracy: 0.6875
```



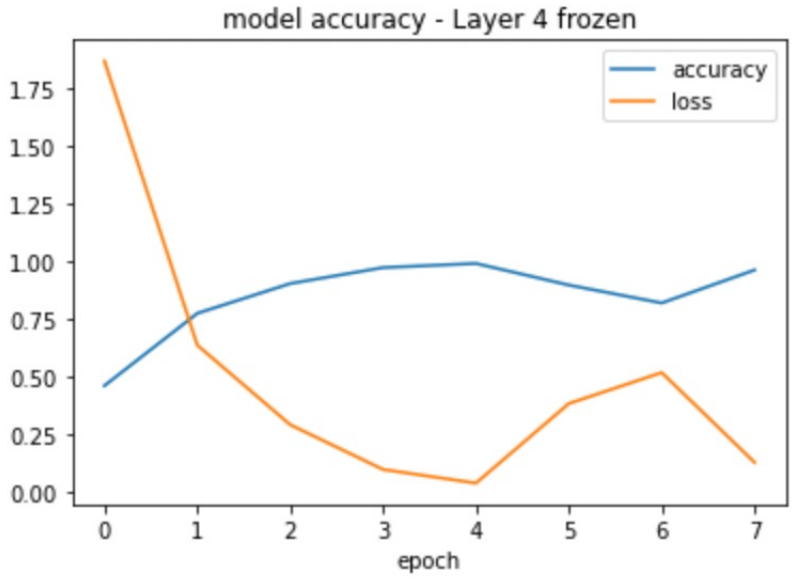
2. Model Accuracy keeping Layer 3 Frozen

```
Test loss: 0.003857143921777606
Test accuracy: 1.0
```



3. Model Accuracy keeping Layer 4 Frozen

```
Test loss: 0.00042600714368745685
Test accuracy: 1.0
```



3 Experimental Setup

Hyperparameters are non-learnable parameters. We experimented with a bunch of different hyper-parameters. We worked with -

1. The number of Hidden Layers: Layers between input and output. We started with 1 Convolution layer and tested the accuracy on test day which was not good. We tried with 2 layers assuming the model might be underfit there was no significant change in the accuracy. After adding 3 layers and changing the parameters defined in the layer like Kernel size, and padding we noticed the accuracy is good. We also use 2 dense layers in our network. We took some ref from standard CNN arch's like AlexNet.

2. Activation Function: We have used the softmax function for the output layer of our network since it's a classification problem and softmax would give us the probabilities of the input belonging to each class. We used relu function in both the convolution and dense layers to achieve good accuracy on

test data.

3. Learning rate: We first used adam for optimizer and then changed it to sgd with a learning rate of 0.01 and noticed that there was some improvement in the accuracy.

4. The number of epochs: We started with 50 in the starting but later changed it to 5 since we have added 3 layers and noticed that it is taking quite a lot of time. At last, we have used 10 epochs to get an accuracy of more than 80%.

5. Batch Size: The batch size was already given to us as 16. We have not made any changes in the batch size.

3.1 Comparing Results

3.1.1 Dataset Configuration

We tried a bunch of different configurations while initialising the ImageDataGenerator to try and improve our results. Such as:

zca_epsilon=1e-06,
rotation_range=0.2,
width_shift_range=0.05,
height_shift_range=0.05,
shear_range=0.05,
zoom_range=0.05,
horizontal_flip=True,
fill_mode='nearest'

We experimented with these image augmentation methods to help our network learn more about the spectral graphs. We were able to achieve more than 80% accuracy without using anything other than 'zca_epsilon' in our final dataset.

While loading the images using the 'flow_from_directory' method we noticed that setting the batch_size and class_mode='categorical' parameters improved our model's performance from 0.867 to 0.892 which was a significant bump.

3.1.2 Algorithm Configuration

This was mostly experimenting with hyperparameters and network design as described above.

We took references from some famous CNN arch's like AlexNet. We tried a single conv layer initially but our test and train accuracy wasn't good enough, so that suggested too few learnable parameters or an underfit. Finally we settled at 3 convolution and 2 dense layers.

From here on we experimented with different hyperparameters on task 1 and used those same parameters for L1 and L2 regularized implementation of our CNN arch.

References

[1] Building CNN in Keras
https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbba
[2] StatQuest
https://www.youtube.com/watch?v=68BZ5f7P94E&list=PLblh5JKOoLUIxGDQs4LFFD--41Vzf-ME1&index=7&ab_channel=StatQuestwithJoshStarmer