THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Comparing Baseline and LeNet for Hand Gesture Recognition

Final Project - Machine Learning 2 (Fall 2020)

Arshiful Islam Shadman
G36335759

Contents

A. Objective
B. Introduction
C. Description of Data
D. Lenet (94)
E. Methodology
F. Results
G. Conclusion

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Objective

The only objective of this project is to look into the performance of two different Convolutional Neural Network architectures: The baseline model and the Lenet (1994) in order to recognize twenty six hand gestures from the American Sign Language which represents the letters of the alphabet.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

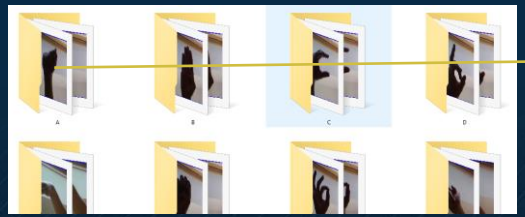
Introduction

Hearing loss, deafness, **hard of hearing**, anacusis, or **hearing** impairment, is defined as a partial or total inability to **hear**. People suffering from this disability tend to communicate with sign languages. The American Sign Language (ASL) is a visual language that includes body movements, facial expression and hand gestures.

Some of these hand gestures represents the letters of the alphabets. In this project the 26 alphabets are being classified using computer vision.

Description of the dataset

The dataset is being downloaded from Kaggle which is a collection of images of alphabets from the American Sign Language arranged in folder marked with their respective labels. Meaning it consists of 29 different folders (i.e. A-Z, Space, Nothing and Delete). Each of this folders contain 3,000 images of each kind summing up to a total of 87,000 images. Therefore the dataset being used does not suffer from imbalances. Each of these images are of 200*200 pixels.



A

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

CNN LeNet5 (1994)

It is a CNN model by Yann LeCun and it insights that image features are distributed across the entire image and that convolutions are an efficient way to extract similar features from multiple locations (parameters sharing).

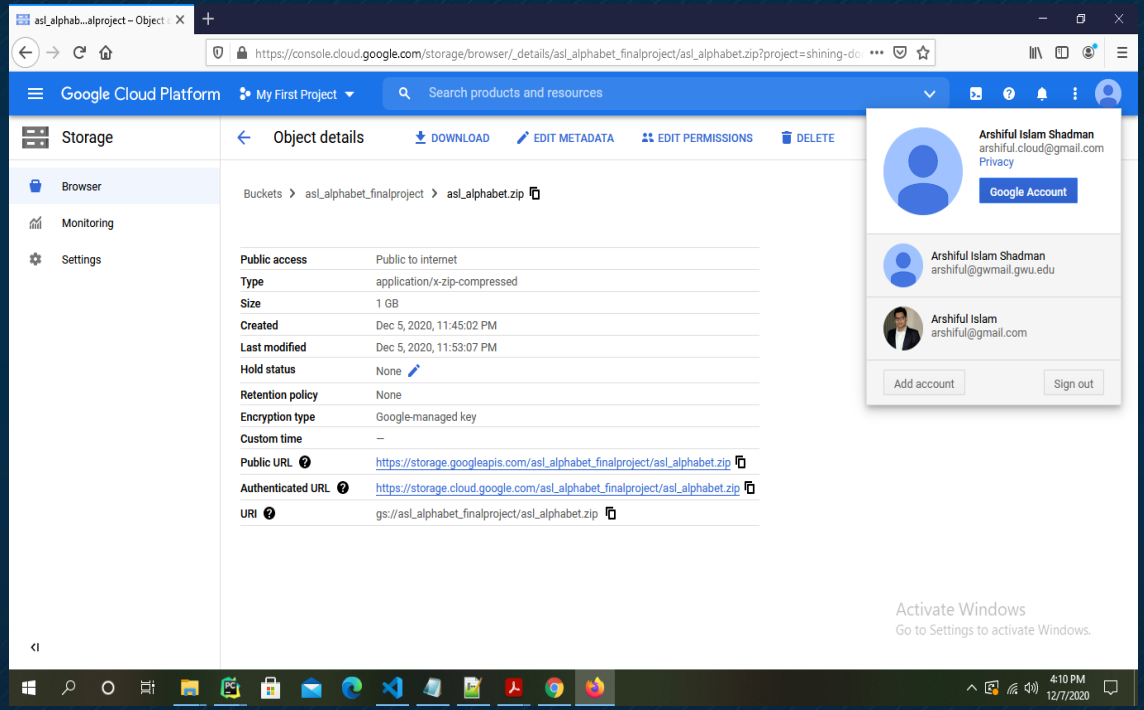
Lenet has much less parameters than conventional fully connected neural networks. Images that are highly spatially correlated and convolutions take advantage of it. And it is specially designed to recognize handwritten digits.

Methodology

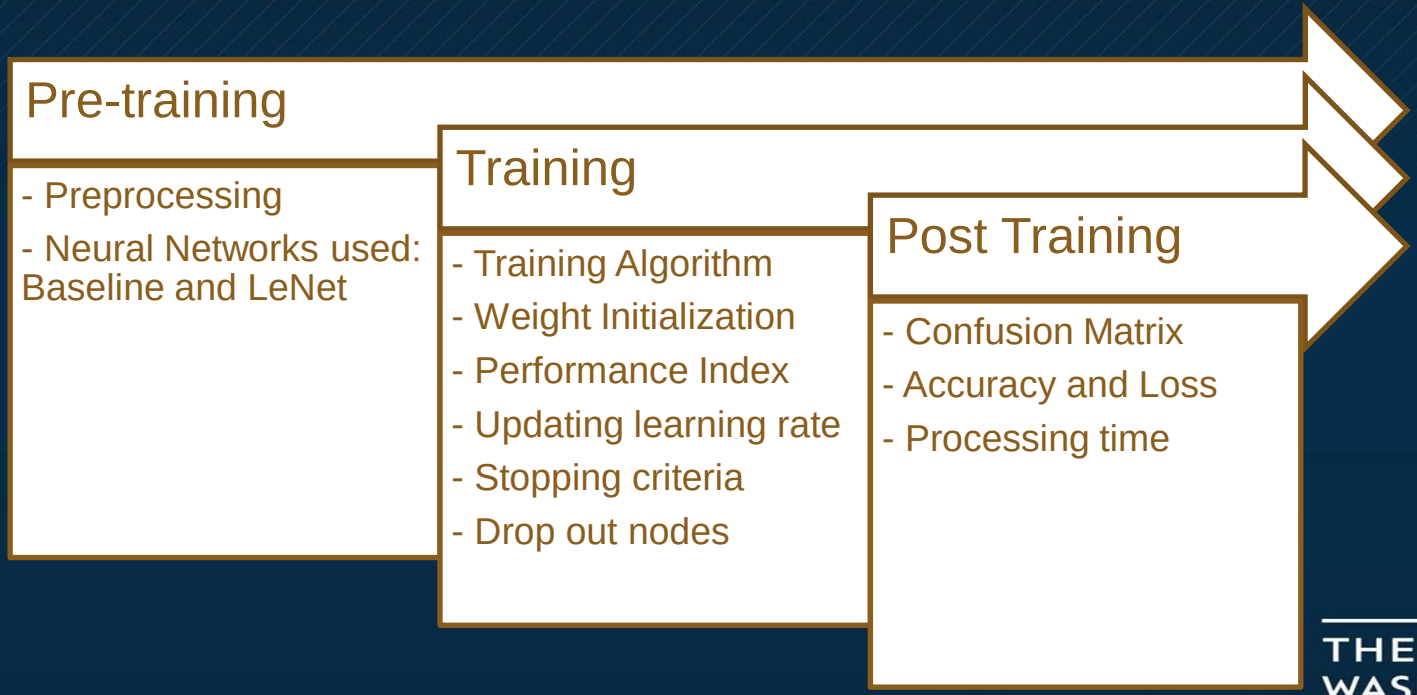
Downloaded the data from Kaggle (URL: https://www.kaggle.com/grassknoted/asl-alphabet)

Created a bucket under the storage section of Google cloud and named it as asl_alphabet_finalproject.

Uploaded the data zipped folder (i.e. asl_alphabet.zip) onto the bucket and made it public (Public URL: https://storage.googleapis.com/asl_alphabet_finalproject/asl_alphabet.zip



Methodology



THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Methodology - Pretraining

Preprocessing:

1. Normalization: **Since its an image we are sending 3 values of mean and 3 values of SD for each color channel RGB, this helps to get the data within a range (-1,1) which helps to train a lot faster.**

```
transform = transforms.Compose([transforms.ToTensor(),transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

2. Train : Validation : Test: **A split of 70 : 15 : 15**

3. No missing values: **3000 images of each type, data pretty balanced**

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Methodology - Pretraining

Preprocessing:
NNs used :

1. Baseline Method
2. LeNet

An issue:

```
ubuntu@image-of-instance-
load_data_p1.py: Started
Killed
ubuntu@image-of-instance-
```

- Running out of memory
- Kept Image of instance same in GCP but customized the RAM
- Could not load all 87000 images in one batch, running out of memory
- Decreased the batch size to 5800

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Methodology - Pretraining

NNs Used:

	Baseline	LeNet
# of Convolutional Layers	2 CONV(5,5)	2 CONV(5,5)
# of fully connected layers	1 (250 neurons)	2 (120,84 neurons)
Transfer function in the output layer	LogSoftMax	Linear

Methodology - Training

Performance Index (Cross Entropy):

Cross-entropy measures the performance of a classification model whose output is a probability value between 0 and 1.It is used to define a loss function, which is to be minimized by updating weights based on the loss function.

Optimizer (Adam):

Adam optimizer is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. It provides an optimization algorithm that can handle sparse gradients on noisy problems and is relatively easy to configure where the default configuration parameters do well on most problems.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Methodology - Training

Weight Initialization (Xavier Normal Initialization):

Xavier Normal Initialization algorithm automatically determines the scale of initialization based on the number of input and output neurons and makes sure that the weights are not too small or too big.

Reduce learning rate on plateau:

It is a scheduler in Pytorch that updates the learning rate based on the value of a specific output (losses or accuracies) when that value is constant after a certain number of epochs.

Methodology - Training

Stopping criteria (Loss trends):

The number of epochs selection is based on the trends between the training and the validation losses

Stopping criteria (Accuracy trends):

In addition to the loss trends the accuracy trend is also used as a stopping criteria

Dropout Nodes:

A default value of (p = 0.5) of dropout nodes during the training.

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Methodology – Post Training

Classic Metric

Confusion matrix

Precision/ Recall/ F-score

AUC/ ROC

Other Metric

Error Analysis

Processing time

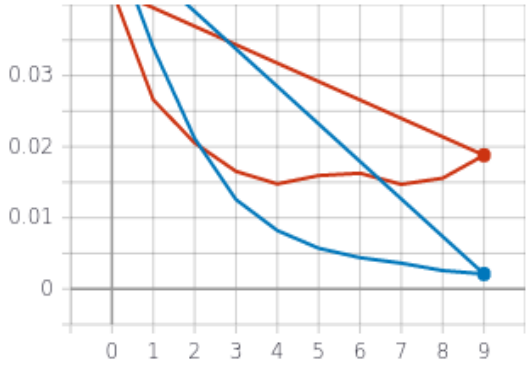
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Results

Baseline Model (Loss and Accuracy):

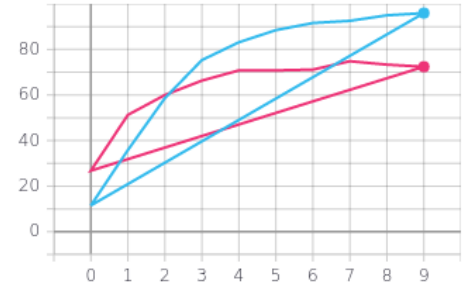
losses

Name	Smoothed	Value	Step	Time	Relative
models/losses/train_bm	2.0949e-3	2.0949e-3	9	Tue Dec 8, 21:38:20	39m 41s
models/losses/val_bm	0.0188	0.0188	9	Tue Dec 8, 21:38:20	39m 41s



accuracies

accuracies



Name	Smoothed	Value	Step	Time	Relative
models/accuracies/train_bm	95.96	95.96	9	Tue Dec 8, 21:38:20	39m 41s
models/accuracies/val_bm	72.41	72.41	9	Tue Dec 8, 21:38:20	39m 41s

losses

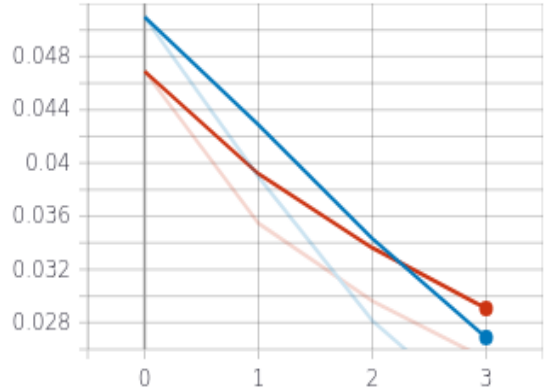
UNIVERSITY
WASHINGTON, DC

Results

LeNet (Loss and Accuracy):

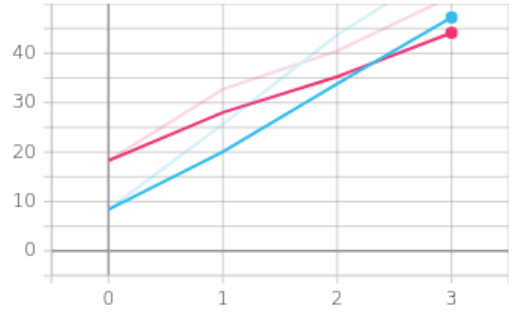
Name	Smoothed	Value	Step	Time	Relative
models/losses/train_ln	0.02688	0.02071	3	Tue Dec 8, 21:33:44	6s
models/losses/val_ln	0.02906	0.02527	3	Tue Dec 8, 21:33:44	6s

losses



accuracies

accuracies



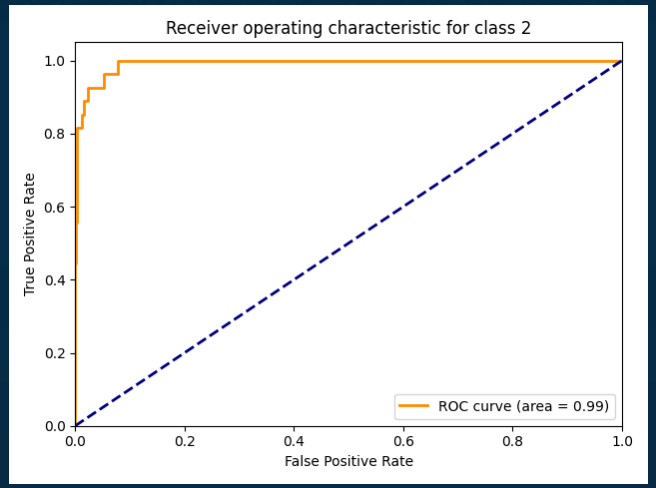
Name	Smoothed	Value	Step	Time	Relative
models/accuracies/train_ln	47.24	58.4	3	Tue Dec 8, 21:33:44	6s
models/accuracies/val_ln	44.13	51.49	3	Tue Dec 8, 21:33:44	6s

losses

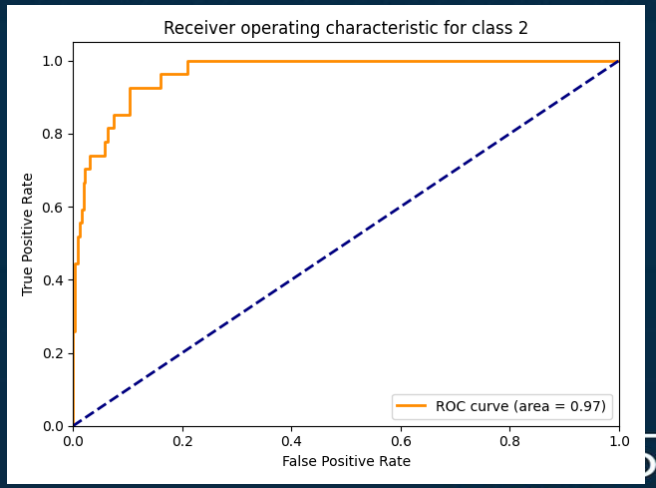
Results

Summary:

Baseline ROC



LeNet ROC



ORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Results

Summary:

	Baseline	LeNet
Precision	0.00807274643571692	0.5143162561498629
Training Accuracy	95.96%	58.39%
Validation Accuracy	72.41%	51.49%
F-Score	0.7209187707367711	0.48049252379868024
AUC	0.99	0.97
Error SD Error Mean	0.00807274643571692 0.02018198514806813	0.00811545001871179 0.03431515967708894

Conclusion

The baseline model seems to perform better than LeNet. However time wise LeNet is faster. The accuracy on the validation is low for LeNet but this is because of the batch size. Changing the number of neuron or increasing the batch size could improve the models significantly.