# DL Assignment 2

| | |
|---|---|
| **Drishya Uniyal** | **MT21119** |
| **Prashant Sharma** | **MT21227** |

**PART I: OPTIMIZERS**
**Implement Optimizers on Multilayer Perceptron from scratch. DO NOT USE DL libraries in PART 1.**
**Use the best result configurations you obtained from Assignment-1 part 3, and replace SGD with the other optimizers discussed below:**
**1. The following optimizers will be implemented from scratch: [4 * 5 = 20]**
**a. Gradient Descent with Momentum**
**b. Nestrov's Accelerated Gradient**
**c. AdaGrad**
**d. RMSProp**
**e. Adam**
Done for Relu + Normal (Dataset Fashion Mnist)

| | **Train Accuracy** | **Val Accuracy** | **Train Loss** | **Val Loss** | **Test Acc** |
|---|---|---|---|---|---|
| **Gradient Descent with Momentum** | 0.9145 | 0.8551 | 0.2394 | 0.4550 | 0.8501 |
| **Nestrov's Accelerated Gradient** | 0.9490 | 0.8683 | 0.1394 | 0.6036 | 0.8650 |
| **AdaGrad** | 0.8566 | 0.8542 | 0.4283 | 0.4353 | 0.8458 |
| **RMSProp** | 0.9684 | 0.8821 | 0.1011 | 0.6856 | 0.8793 |
| **Adam** | 0.9589 | **0.8826** | 5.97 | 6.22 | **0.8846** |

**2. For each optimizer you try above, generate the following plots: [5 marks]**

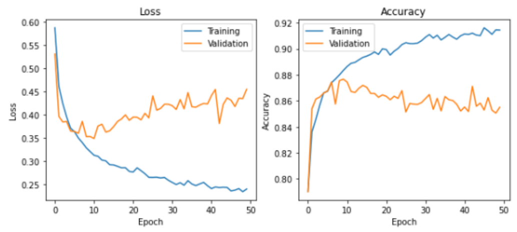
**a. Loss plot - Training Loss and Validation Loss V/s Epochs.**

**b. Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs)**

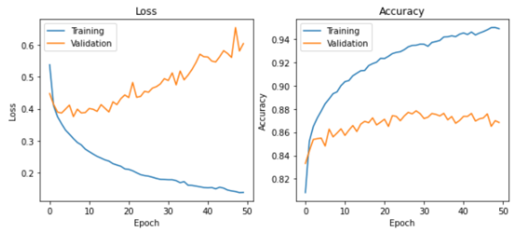
**c. Analyze and Explain the plots obtained.**

## <u>Loss and Accuracy Plots</u>

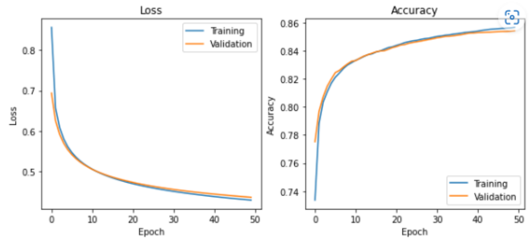
### Gradient Descent with Momentum



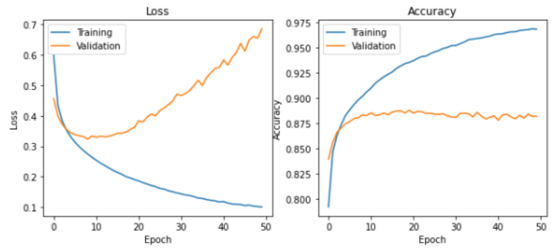
### Nestrov's Accelerated Gradient



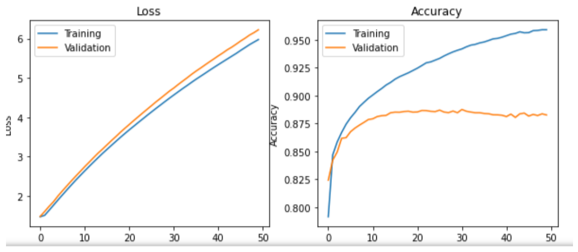
### AdaGrad



### RMSProp



### Adam

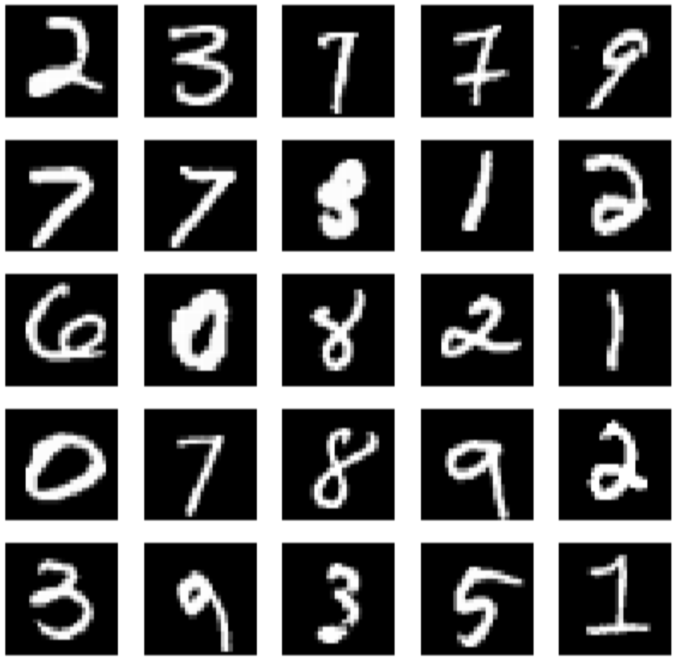


**Analysis:**

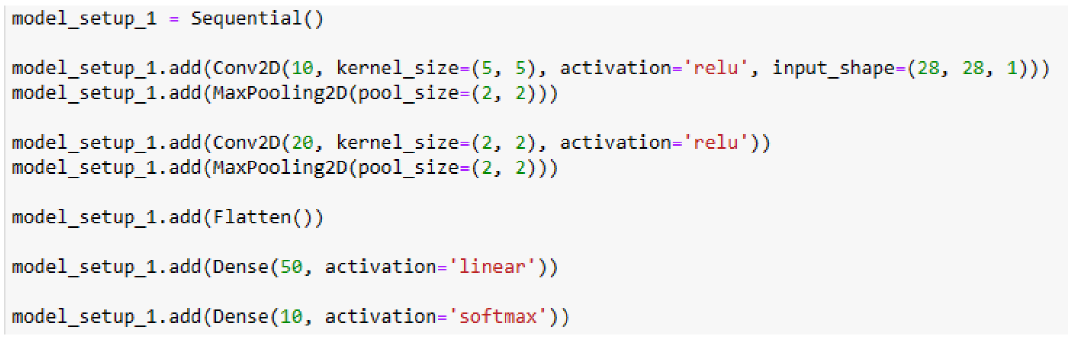
1. Parameters: Epochs - 50, Learning Rate - 0.001, momentum - 0.9, beta1=0.9, beta2=0.999, epsilon=1e-8, l2_reg=0.01
2. The plots suggest train loss, val loss, and train acc and val acc, respectively.
3. Regularization has been done.
4. The plots suggest Adam performed best, but the loss is enormous.
5. Gradient Descent and Nesterov performed almost equally.

**For PART 2 and 3, you can use DL libraries of your choice.**
**PART II: Convolution Neural Network (CNN) (20 marks)**
**Dataset: MNIST**
**If you are using PyTorch, then download the dataset from torchvision.datasets library, and if you use**
**tensorflow, then use tensorflow.keras.datasets to load the data. Don't download the data from any**
**other source, as the train test split on different links may vary.**

**Dataset description: The dataset contains 60,000 images in the training set and 10,000 in the test set. Each image is of 28 X 28 size digits between 0 to 9.**
**Task: Develop a neural network to identify the digits. Perform splitting of Train data in training, validation sets with an 80:20 ratio (random stratified) and use the test data from the test.csv file.**
**1. Visualise 5 random images from 5 different digits. [5 marks]**
**Results:**



**2. Setup1: Create a CNN architecture having [4*2 = 8 marks]**
**a. a kernel size of 5×5, followed by kernel of 2x2. Use 10 feature maps for the first convolution layer and 20 for the second.**
**b. Add pooling after each co nvolution.**
**c. Add a linear layer with 50 neurons. Finally, add a classification head. Use softmax activation function at classification head.**
**d. In the above setup you are free to test varying pooling techniques, LR rates, optimisers and activation functions, strides and padding.**



```
model_setup_1 = Sequential()

model_setup_1.add(Conv2D(10, kernel_size=(5, 5), activation='relu', input_shape=(28, 28, 1)))
model_setup_1.add(MaxPooling2D(pool_size=(2, 2)))

model_setup_1.add(Conv2D(20, kernel_size=(2, 2), activation='relu'))
model_setup_1.add(MaxPooling2D(pool_size=(2, 2)))

model_setup_1.add(Flatten())

model_setup_1.add(Dense(50, activation='linear'))

model_setup_1.add(Dense(10, activation='softmax'))
```

We try different activations and optimizers for the above models to check the best results.

**Table:**
**Epochs = 30**
**Loss = categorical_crossentropy**

| **Activation** | **Optimizer** | **Train Acc** | **Val Acc** | **Test Acc** |
|---|---|---|---|---|
| Relu | Adam | 99.29 | 97.73 | 98.14 |
| Sigmoid | Adam | 99.08 | 97.90 | 98.15 |
| Relu | Rmsprop | 99.42 | 98.13 | 98.39 |
| **Sigmoid** | **Rmsprop** | **99.04** | **98.11** | **98.51** |

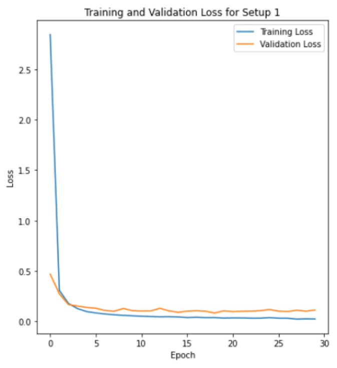
**3. Setup 2: Update setup 1 by adding a residual connection after one of the layers as you deem fit. [2 marks]**
Taking the best setup from above to perform on this part.

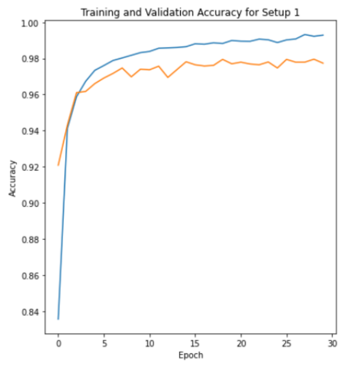
| **Train Acc** | **Val Acc** | **Test Acc** |
|---|---|---|
| **99.50** | **98.38** | **98.40** |

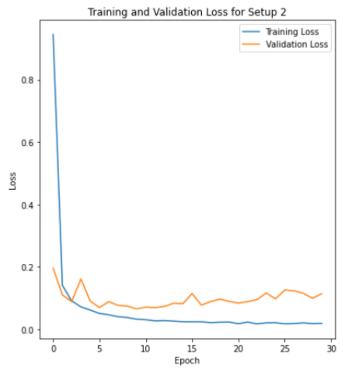
The result shows an improvement in accuracy.

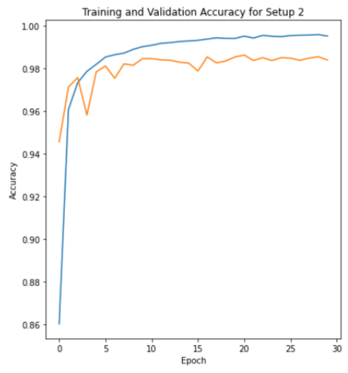
**4. Generate the following plots for each setup: [5 marks]**

○ **Loss plot - Training Loss and Validation Loss V/s Epochs.**
○ **Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs**









○ **Analyze and Explain the plots obtained**
Results:

1. The graphs above are plotted for loss and accuracy for the best model achieved!
2. In the first setup, the validation loss gradually decreases and remains almost constant, while in setup two, it fluctuates and goes a bit high.
3.

| | | |
|---|---|---|
| Setup 1 | Train - 0.0290 | Val - 0.0678 |
| Setup 2 | Train - 0.0181 | Val - 0.1134 |

4. The accuracy in the second setup increased, which can be seen by the graph. It initially starts with higher accuracy.

**PART III: Data Augmentation(15 marks)**

**Use the best model configuration from Part 2, and try each of the below augmentations one by**
**one on a random stratified subset 20% of the train split. Add the modifications to your original train set and observe the change in performance (one augmentation at a time).**
**NOTE: DO NOT PERFORM ANY AUGMENTATION ON THE VAL OR TEST SET.**
**1. Positional Augmentation: [2*5 = 10]**
**a. Resize your data. [2 marks]**

| Train Acc | Train Loss | Val Acc | Val Loss |
|---|---|---|---|
| 0.9786 | 0.0687 | 0.9758 | 0.0838 |

**b. Left-right flip the original data. [2 marks]**

| Train Acc | Train Loss | Val Acc | Val Loss |
|---|---|---|---|
| 0.9602 | 0.1245 | 0.9682 | 0.1027 |

**c. Rotate the original data by some degree. [2 marks]**

| Train Acc | Train Loss | Val Acc | Val Loss |
|---|---|---|---|
| 0.9834 | 0.0512 | 0.9809 | 0.0692 |

**d. Add some Gaussian noise to the original data. [2 marks]**

| Train Acc | Train Loss | Val Acc | Val Loss |
|---|---|---|---|
| 0.9898 | 0.0313 | 0.9833 | 0.0650 |

**e. Combine all the augmentation steps among the above four parts. [2 marks]**

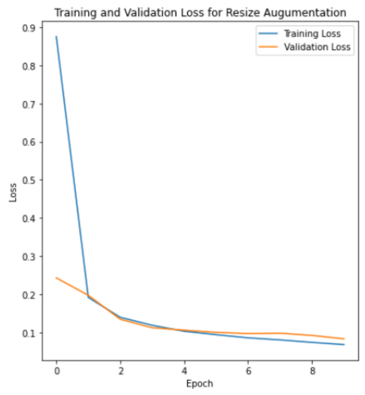
| Train Acc | Train Loss | Val Acc | Val Loss |
|---|---|---|---|
| 0.9687 | 0.0960 | 0.9750 | 0.0833 |

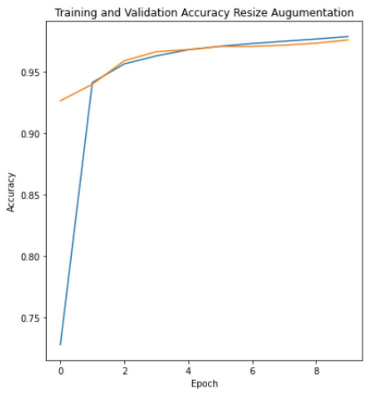
Sol: Different Functions have been made for the above parts!
Performed on the best model obtained from part 2.

**2. Generate the following plots for each augmentation: [5 marks]**
**○ Loss plot - Training Loss and Validation Loss V/s Epochs.**
**○ Accuracy plot - Training Accuracy, Validation Accuracy V/s Epochs**
**○ Analyze and Explain the plots obtained, especially the role different augmentation played in improving the accuracy, if any.**

**Resize**

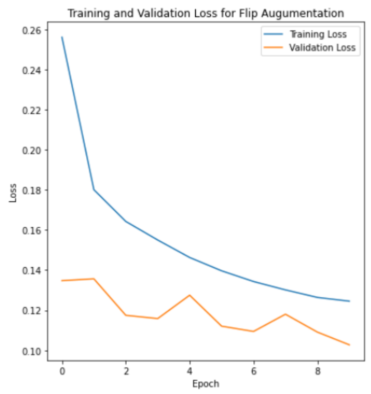
The plot shows decreasing loss, and accuracy increases gradually but lesser than the original model.

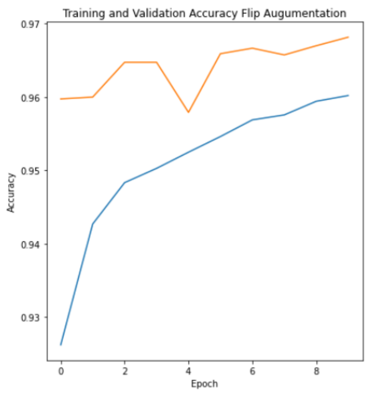




**Left-right flip the original data**

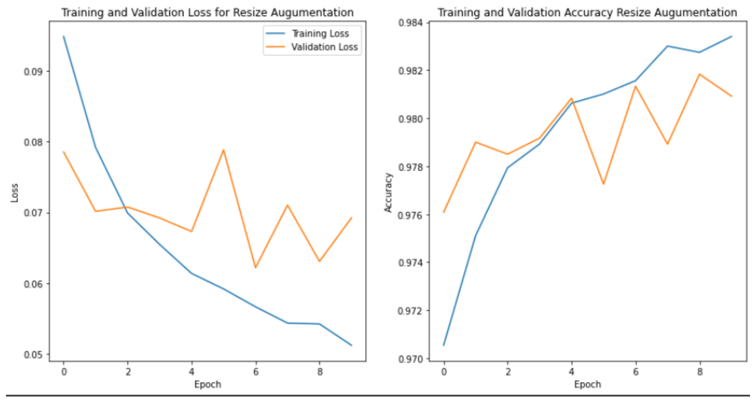
Here, the graph shows less training accuracy and more validation accuracy, which mainly comes from the given data as we have taken only 20% of the training data, so increasing the data and then flipping might increase the accuracy





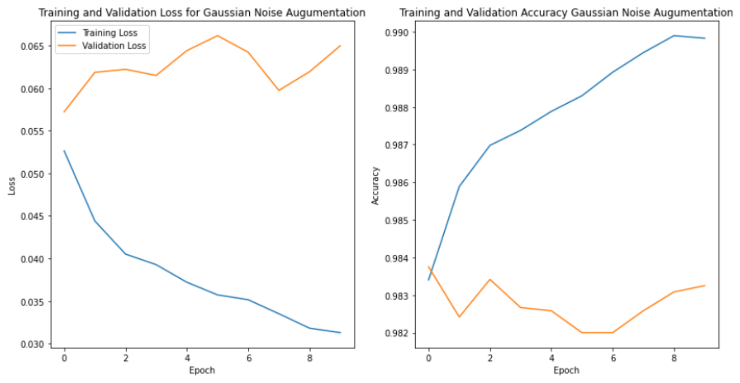
**Rotate the original data by some degree**

Rotating the image increases the data, and so the training accuracy is higher than the validation accuracy as the rotation was done on train data. The graph shows higher train accuracy and lesser train loss.



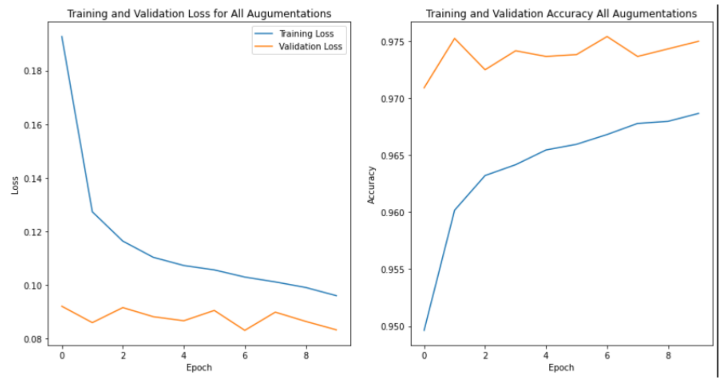
**Add some Gaussian noise to the original data**

The graph shows that the validation accuracy is significantly less and also loss is higher, which is terrible. To avoid overfitting, we use this, but if the noise added is too large, then it reduces the quality of the data and then tends to overfit, the case here!!



**Combine all the augmentation steps among the above four parts**

Here, the graph shows higher validation accuracy means the model performs better and loss is also less. Combining all improves the overall accuracy but still less than the original model.



**Contributions:**

Drishya:- Part 1 and Part of Par 3

Prashant:- Part 2 and Part 3

Report contribution equal