Deep Learning

**MULTIMODAL CLASSIFICATION**

ASSIGNMENT 2

horizontal line

# FC Neural Network

### Hyperparameters

* 1. Learning rate (0.01, 0.001)
  2. Optimizer (sgd, adagrad, adam, rmsprop)
  3. Loss
     1. Only the sparse categorical cross entropy works as the output is digit and not 1 hot encoding, else we should have used categorical cross entropy.
  4. Activations functions (Sigmoid, Relu, Tanh)
     1. The activation function for the output layer must be softmax

### (Using default LR and relu activation function) [Optimizer HP]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimizer | SGD | Adagrad | RMSprop | Adam |
| Train-Acc | 0.91 | 0.91 | 0.981 | 0.983 |
| Test-Acc | 0.92 | 0.91 | 0.975 | 0.976 |
| Graph | [sgd\_opt](#_crtb0pdatuk8) | [adagrad\_opt](#_91zac4h9fgs1) | [rmsprop\_opt](#_d46kqhgyaxpo) | [adam\_opt](#_7vt4evpj8lqg) |

### (Using relu activation function) [Learning rate HP]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimizer | RMSprop | | Adam | |
| LR | 0.01 | 0.01 | 0.01 | 0.001 |
| Train-Acc | 0.978 | 0.981 | 0.975 | 0.983 |
| Test-Acc | 0.967 | 0.975 | 0.963 | 0.974 |
| Graph | [rmsprop\_0.01](#_1lu50rovvqc) | [rmsprop\_0.001](#_l5f3alx8ooz) | [adam\_0.01](#_t3hx2uzht20n) | [adam\_0.001](#_dhpgb563jjty) |

### (Using Adam and LR = 0.001 ) [Activation HP]

|  |  |  |  |
| --- | --- | --- | --- |
| Activation | Sigmoid | Relu | tanh |
| Train-Acc | 0.966 | 0.982 | 0.981 |
| Test-Acc | 0.964 | 0.976 | 0.973 |
| Graph | [sigmoid\_act](#_p200om5c2uyw) | [relu\_act](#_9kufrdebug8l) | [tanh\_act](#_3u62sn4rmos2) |

### Analysis

* 1. Best optimizer: Adam and RMSprop with Adam slightly better
  2. Best Learning rate: 0.001 with both Adam and RMSprop
  3. Best Activation Function considering the best optimizer with best LR: Relu slightly better than tanh.

### Observation

* 1. Test accuracy never reached the training accuracy
  2. Overfitting happens some of the time

# Convolution Neural Network

### Hyperparameters

* 1. Activation (Relu, Tanh)
  2. Filter size (3, 5, 7)
  3. Stride length (1, 2, 4)
  4. Pool layers (2, 3, 4)
  5. Dropout (hidden/input - 0.2/0.4)

### Fixing other parameters to see best Activation (3x3, 1, 2 maxpool)

|  |  |  |
| --- | --- | --- |
| Activation | Relu | Tanh |
| Train-Acc | 0.994 | 0.991 |
| Test-Acc | 0.991 | 0.986 |
| Graph | [Relu\_act](#_l17c9herws2) | [Tanh\_act](#_2i14vqxa2qqj) |

### Fixing other parameters to see best Filter size and stride (relu, 2 maxpool)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Size | 3x3 | | | 5x5 | | | 7x7 | | |
| Stride | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| Train-Acc | 0.994 | 0.971 | 0.969 | 0.994 | 0.988 | 0.981 | 0.991 | 0.978 | 0.983 |
| Test-Acc | 0.988 | 0.97 | 0.965 | 0.991 | 0.984 | 0.98 | 0.986 | 0.972 | 0.981 |
| Graph | [fig\_1](#_3hcr95t5fvh5) | [fig\_2](#_v42kzb13fqmy) | [fig\_3](#_2b57qjftb0lq) | [fig\_4](#_abcipji7i39v) | [fig\_5](#_naxhwy68ggbm) | [fig\_6](#_2wv1mrrcowgm) | [fig\_7](#_urpaxcqh80lr) | [fig\_8](#_bh6pbhcynbiv) | [fig\_9](#_wr2tmnk1kl8o) |

#### Observation:

Decreasing the stride length always result in better accuracy and the opposite  
 happens with the filter size, as size increase the accuracy increases till an overfitting  
 happens

### Fixing other parameters to se best num of maxpool layers

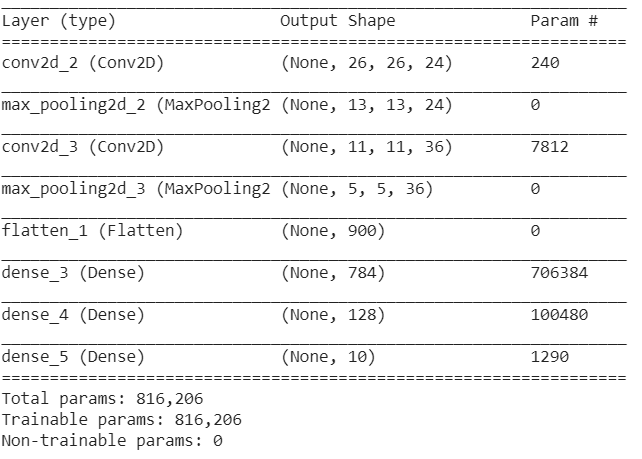
|  |  |  |  |
| --- | --- | --- | --- |
| Maxpool num | 2 | 3 | 4 |
| Train-Acc | 0.994 | 0.99/0.992 | 0.985 |
| Test-Acc | 0.989 | 0.988/0.989 | 0.985 |

### 

### Fixing other parameters and using dropout as HP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position | Hidden | | Input | |
| prob | 0.2 | 0.4 | 0.2 | 0.4 |
| Train-Acc | 0.99 | 0.98 | 0..99 | 0.986 |
| Test-Acc | 0.98/0.99 | 0.99 | 0.987 | 0.984 |

### Model Analysis

Analysis is done for one Model, however it could be repeated to all similarly  


* 1. Running time = 26 sec
  2. Number of parameters = 816,206
  3. Number of multiplications
     1. Training = 60,000 \* ( (26\*26\*24\*3\*3\*1) + (0) + (11\*11\*36\*3\*3\*24) + (0) + (784\*900) + (784\*128) + (128\*10)) \* 5
     2. Testing = 10,000 \* ( (26\*26\*24\*3\*3\*1) + (0) + (11\*11\*36\*3\*3\*24) + (0) + (784\*900) + (784\*128) + (128\*10))

### Comparison to FC NN

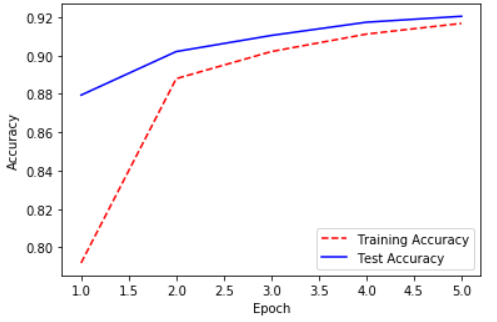
|  |  |  |
| --- | --- | --- |
| Model | FC-NN | CNN |
| Train Accuracy | 0.982 | 0.994 |
| Test Accuracy | 0.974 | 0.991 |
| Time | 23.46 sec | 26.0 sec |

# Appendix

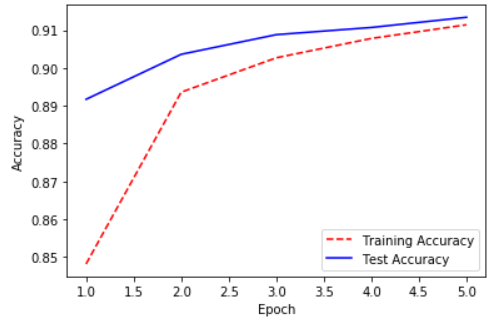
### FC NN

## Optimizers

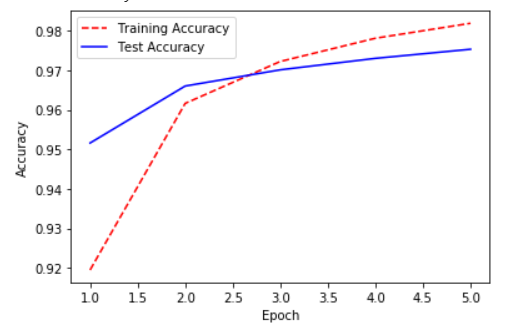
#### Sgd



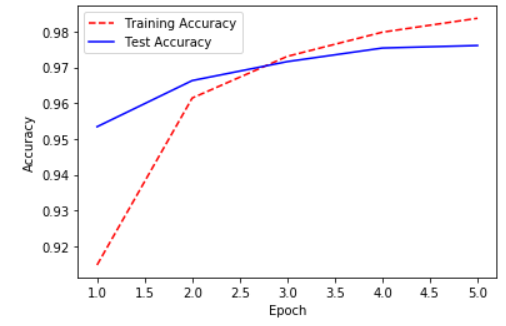
#### Adagrad



#### Rmsprop

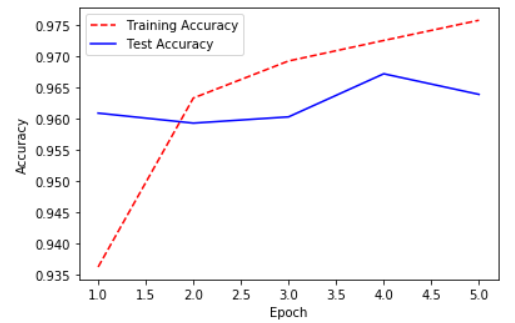


#### Adam

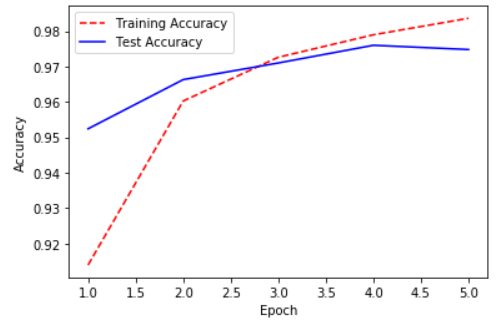


## LR

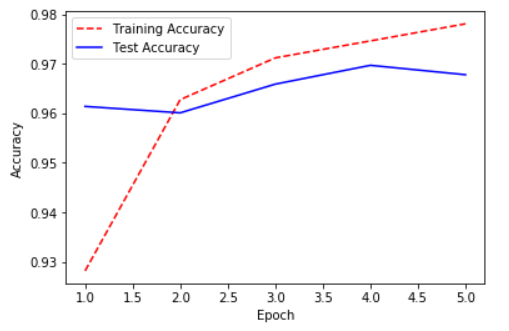
#### Adam\_0.01



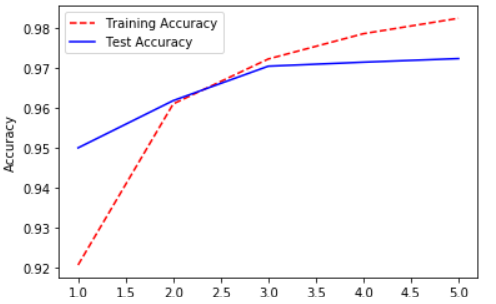
#### Adam\_0.001



#### RMSprop\_0.01

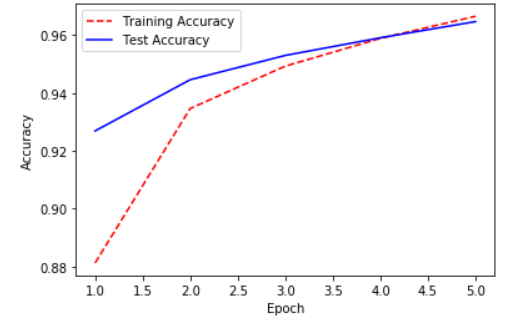


#### RMSprop\_0.001

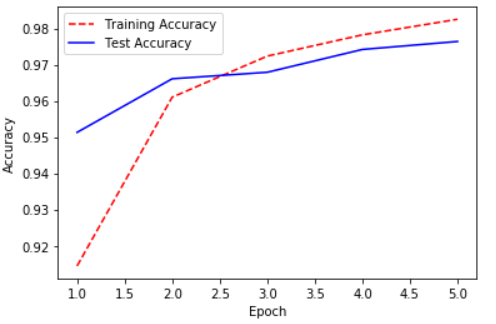


## Activation

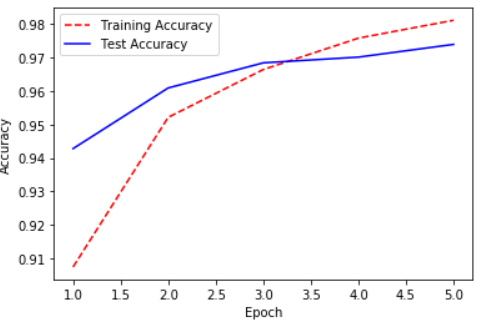
#### Sigmoid



#### Relu



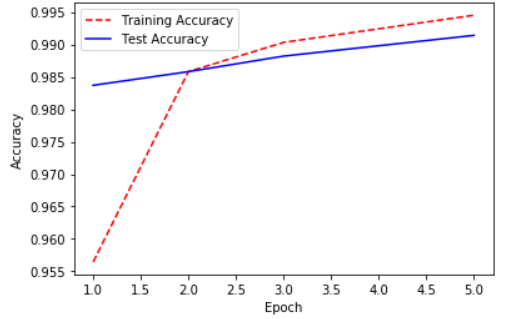
#### Tanh



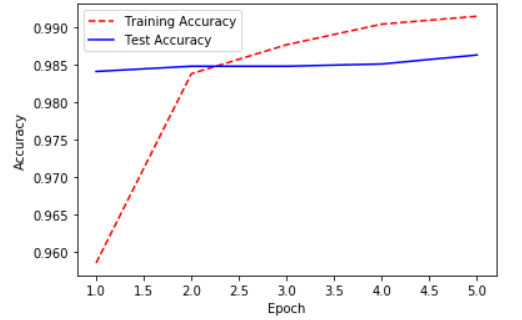
### CNN

## Activation

#### Relu

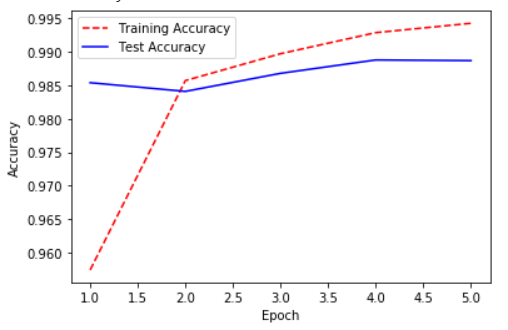


#### Tanh

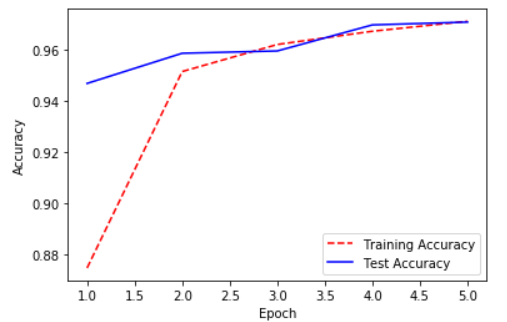


## Filter size & Stride

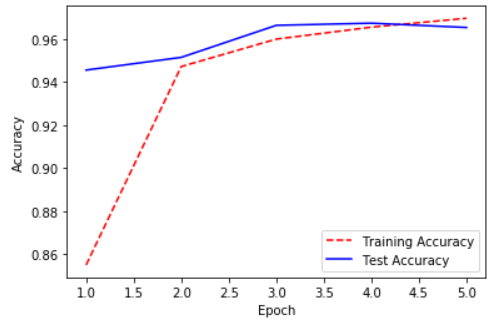
#### 3x3\_1



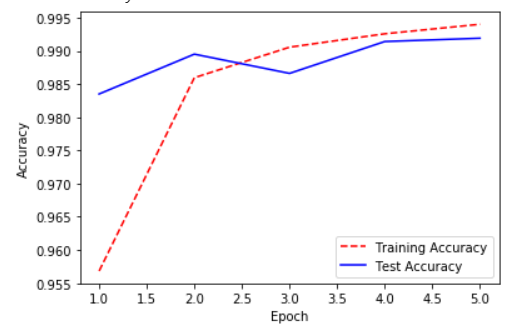
#### 3x3\_2



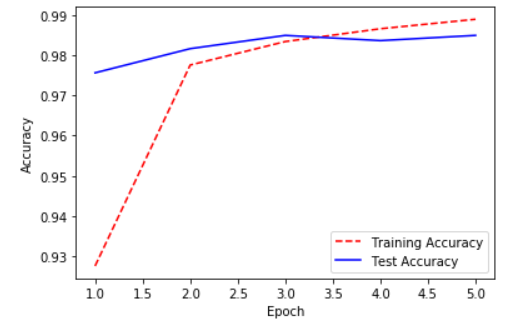
#### 3x3\_3



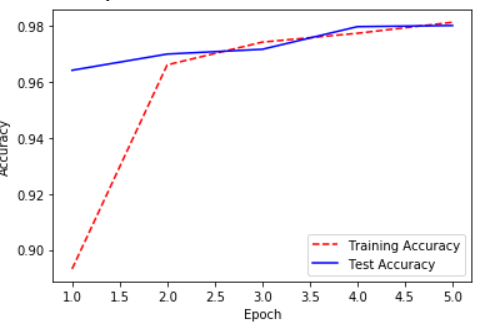
#### 5x5\_1



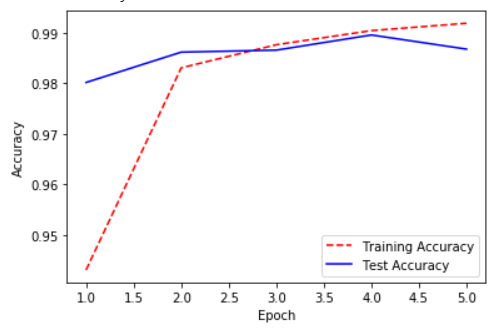
#### 5x5\_2



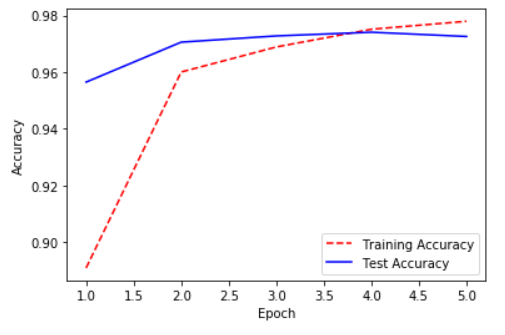
#### 5x5\_3



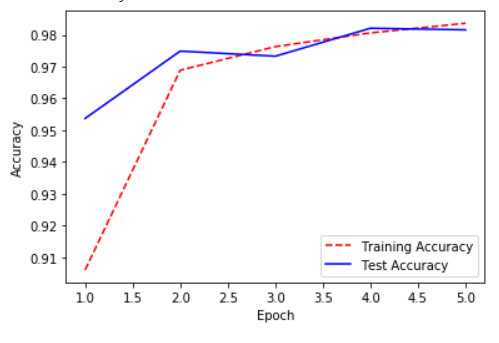
#### 7x7\_1



#### 7x7\_2



#### 7x7\_3



* 1. .