Deep Learning

**CNN AND TRANSFER LEARNING**

ASSIGNMENT 3

horizontal line

# CNN

### Hyperparameters

* 1. Learning rate
     1. 0.01, 0.001
  2. Optimizer
     1. sgd, adagrad, adam, rmsprop
  3. CNN layers
     1. 1, 2, 3
  4. Pooling layers
     1. 1, 2, 3
  5. Pooling type
     1. AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D, GlobalAveragePooling2D
  6. Regularization
     1. Early stopping, l1, l2, data augmentation

### Optimizer and LR HP (using best model found in step 3)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Optimizer | SGD | | Adagrad | | RMSprop | | Adam | |
| LR | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 |
| Train-Acc | 0.97 | 0.96 | 0.98 | 0.98 | 0.95 | 0.97 | 0.97 | 0.98 |
| Test-Acc | 0.91 | 0.93 | 0.95 | 0.94 | 0.90 | 0.94 | 0.90 | 0.95 |

### CNN and pooling layers HP

CNN → 7x7 filters with 1x1 stride length and same padding for first layer  
MaxPool → 2x2 with 1x1 stride length  
Using Batch normalizer for (2 CNN, 1 MP) (2 CNN, 2 MP) (3 CNN, 3 MP)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CNN Layers | 1 CNN | | 2 CNN | | 3 CNN | |
| Pooling Layers | 1 | 2 | 1 | 2 | 2 | 3 |
| Train-Acc | 0.94 | 0.92 | 0.99 | 0.98 | 0.91 | 0.97 |
| Test-Acc | 0.91 | 0.86 | 0.953 | 0.93 | 0.90 | 0.95 |
| Model | [m\_1](#_w91ejebhpf6) | [m\_2](#_xe79azt4wxeq) | [m\_3](#_41e5x1neuebp) | [m\_4](#_a9c3p2ca54nv) | [m\_5](#_ew4fn4a74mqh) | [m\_6](#_47yp9yxdvyyv) |

### Pooling type HP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pooling Type | Avg | Max | Glob\_Avg | Glob\_Max |
| Train-Acc | 0.96 | 0.98 | - | - |
| Test-Acc | 0.91 | 0.953 | - | - |

### Regularization HP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reg | None | ES | L1 | L2 |
| Train-Acc | 0.98 | 0.96 | 0.95 | 0.97 |
| Test-Acc | 0.953 | 0.94 | 0.91 | 0.92 |

### Observation

* 1. Best Model
     1. Optimizer Adam
     2. LR 0.001
     3. No regularization
     4. Use Batch normalization
     5. 2 CNN layers and 1 MaxPool layer
  2. Training accuracy is always higher than test accuracy
  3. Test accuracy is varying due to small epoch
  4. Best Accuracy is 0.95

# ResNet & VGG

### Untrained

#### Hyperparameters

#### Optimizer

* + - 1. Adam, SGD

#### LR

* + - 1. 0.01, 0.001

#### Regularization

* + - 1. Early stopping, l1, l2 (choose random Conv2D layers to apply kernel regularization)
    1. Pooling
       1. Average, max, none
    2. Epochs
       1. 10, 20, 40
    3. Top-layer
       1. Adding dense (512, 1024) with dropout
       2. Adding dense (128, 512, 1024) without dropout

#### Hyperparameter tuning

(Using **SGD**(0.0001, 0.9 mom) and dense layer 128 neuron))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ResNet | | | | | | |
| Epochs | 10 | | 20 | | 40 | |
| Pooling | max | avg | max | avg | max | avg |
| Train-Acc | 0.85 | 0.85 | 0.90 | 0.91 | 0.87 | 0.96 |
| Test-Acc | 0.64 | 0.56 | 0.86 | 0.9 | 0.91 | 0.9 |

(Using **Adam**(0.001) and dense layer 128 neuron))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ResNet | | | | | | | | | | | | |
| Epochs | 10 | | | | 20 | | | | 40 | | | |
| Pooling | max | | avg | | max | | avg | | max | | avg | |
| LR | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 | 0.001 | 0.01 |
| Train-Acc | 0.96 | 0.59 | 0.98 | 0.51 | 0.82 | 0.52 | 0.95 | 0.52 | 0.96 | 0.52 | 0.96 | 0.52 |
| Test-Acc | 0.46 | 0.58 | 0.56 | 0.54 | 0.49 | 0.56 | 0.94 | 0.56 | 0.90 | 0.56 | 0.87 | 0.56 |

(Using Avg pooling and 20 epochs)

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ResNet | | | | | | | | | | | | |
| Dnese | 128 | | | | 512 | | | | 1024 | | | |
| Dropout | No | | 0.5 | | No | | 0.5 | | No | | 0.5 | |
| Opt | Adam | SGD | Adam | SGD | Adam | SGD | Adam | SGD | Adam | SGD | Adam | SGD |
| Train-Acc | 0.93 | 0.89 | 0.72 | 0.86 | 0.88 | 0.93 | 0.95 | 0.89 | 0.95 | 0.91 | 0.97 | 0.87 |
| Test-Acc | 0.80 | 0.82 | 0.88 | 0.86 | 0.84 | 0.84 | 0.93 | 0.85 | 0.89 | 0.82 | 0.88 | 0.84 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | ResNet152 | | | |
| Optimizer | Adam | | | |
| Dense | 512 | | | |
| LR | 0.001 | | | |
| Pooling | Average | | | |
| Epochs | 20 | | | |
| Dropout | 0.5 | | | |
| Reg | None | ES | l1 | l2 |
| Train-Acc | 0.97 | 0.91 | 0.86 | 0.96 |
| Test-ACC | 0.91 | 0.54 | 0.57 | 0.92 |
| Training Time | 79 sec | 58 sec | 79 sec | 79 sec |
| TT per epoch | 3 sec | 3 sec | 3 sec | 3 sec |
| Test time | 0.34 sec | 0.35 sec | 0.37 sec | 0.34 sec |
| Multiplication | Takes too much time to compute | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | VGG16 | | | | | | | |
| Optimizer | Adam | | | | SGD | | | |
| LR | 0.001 | | 0.0001 | | 0.0001 | | 0.00001 | |
| Reg | L2 | Drop | L2 | Drop | L2 | Drop | L2 | Drop |
| Train-Acc | 0.72 | 0.78 | 0.96 | 0.97 | 0.80 | 0.66 | 0.85 | 0.62 |
| Test-ACC | 0.56 | 0.64 | 0.68 | 0.96 | 0.71 | 0.67 | 0.82 | 0.70 |
| Training Time | 23 sec | | | | | | | |
| TT per epoch | 1 sec | | | | | | | |
| Test time | 0.1 sec | 0.1 sec | 0.12 sec | 0.19 sec | 0.16 sec | 0.11 sec | 0.1 sec | 0.1 sec |
| Multiplication | (64\*64\*64\*3\*3\*3)+(32\*32\*128\*3\*3\*64)+(16\*16\*256\*3\*3\*128)+  (8\*8\*512\*3\*3\*256)+(4\*4\*512\*3\*3\*512)+(2\*2\*512\*3\*3\*512)+ (2048\*4096)+(4096\*4096)+4096 | | | | | | | |

### Pretrained HP tuning

* 1. Number of freezed layers
  2. Epochs
  3. Number of dense layers (1 or 2)
  4. Neurons in dense layer (512, 512 and 256)
  5. Optimizers (RMSprop, Adam, SGD)
  6. Learning rate (0.001, 0.0001)
  7. Regularizers (L1, L2, Dropout)

### Best result achieved for:

* 1. ResNet : (opt-Adam, LR-0.001, Reg-L2/no reg, dense layers 2 (512, 256), epochs 35, no freezed) most stable highest accuracy -> 0.94/0.86 followed by 20 layer freezing with 0.90 (by trying 5,15, 20,30,60,100)

**There is excessive randomness during runs I don’t know why, same combinations gave different outputs when run repeatedly**

[Layer -Freezing plot](#_3gqiehb5ian)

[Accuracy Without freezing](#_bb4yqn9i28vx)

* 1. VGG: (opt-Adam, LR-0.0001, no reg, dense layers 2 (512, 256), epochs 25, no freezed) most stable highest accuracy -> 0.973 followed by 5 layer freezing with 0.98 (by trying 1,2,5,10,15)

[Layers-Freezing plot](#_4qux581jatpc)

[Accuracy Without freezing](#_27w8kw9khp94)

[Accuracy with freezing](#_a5u5kh4gxja0)

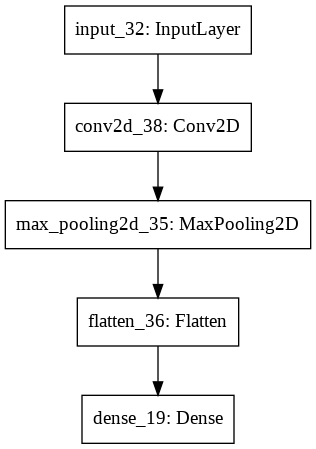
# Appendix

### CNN

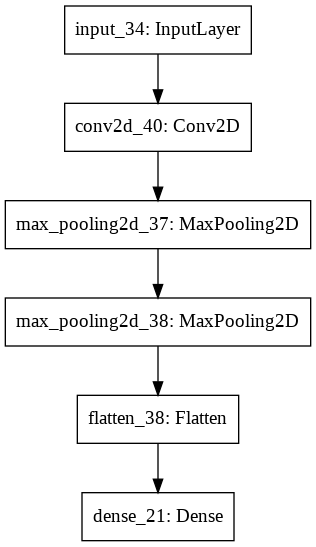
#### LR and Optimizer tuning

#### Model CNN and MP layers tuning

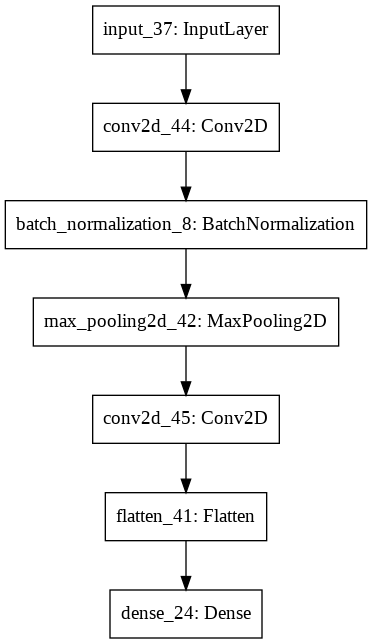
##### M\_1



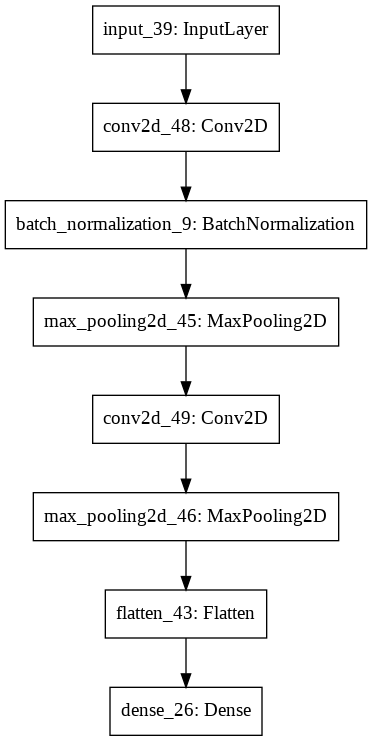
##### M\_2



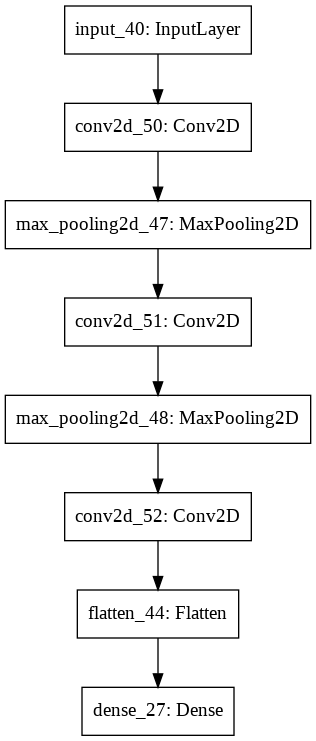
##### M\_3



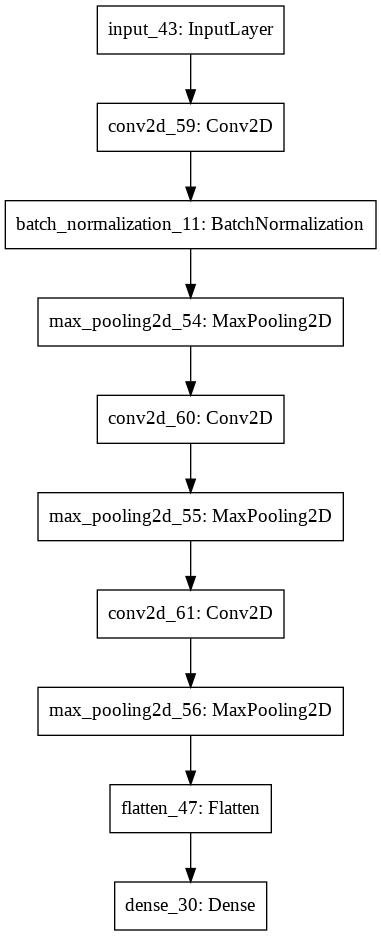
##### M\_4



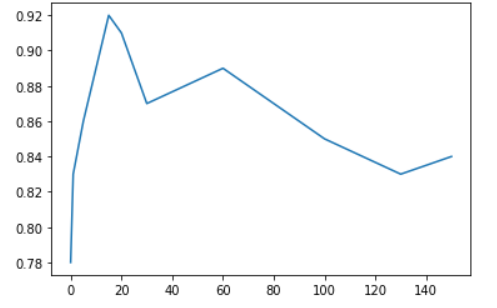
##### M\_5



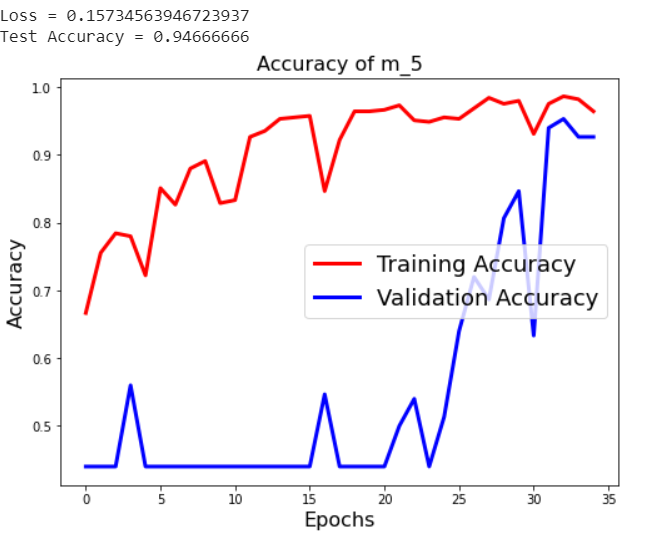
##### M\_6



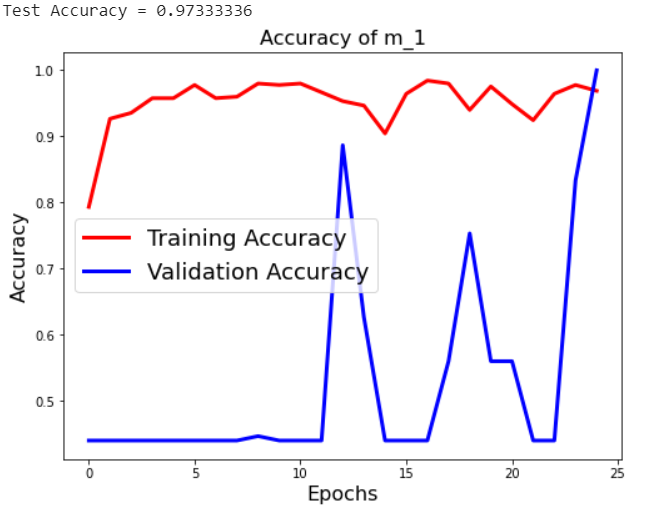
#### Freezing layer plot ResNet



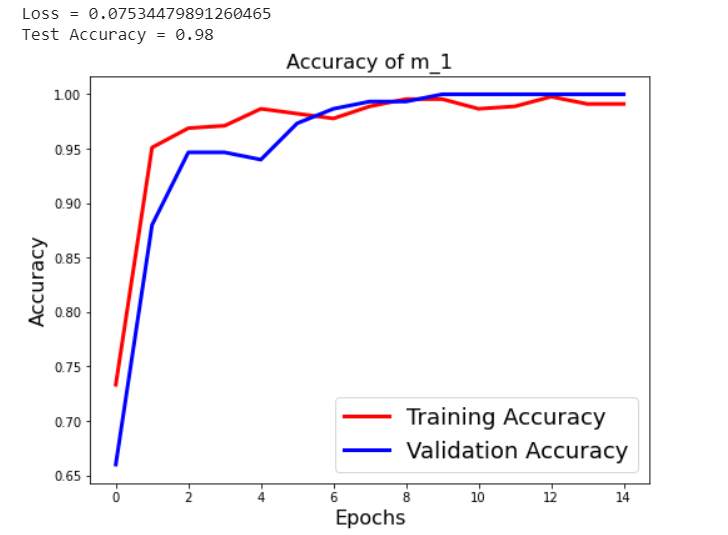
#### Without freezing Resnet



#### Without freezing VGG



#### With freezing VGG



#### Freezing layer plot VGG

