Deep Learning

CNN AND TRANSFER LEARNING
ASSIGNMENT 3

CNN

1. Hyperparameters

a. Learning rate
   i. 0.01, 0.001

b. Optimizer
   i. sgd, adagrad, adam, rmsprop

c. CNN layers
   i. 1, 2, 3

d. Pooling layers
   i. 1, 2, 3

e. Pooling type
   i. AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D, GlobalAveragePooling2D

f. Regularization
   i. Early stopping, l1, l2, data augmentation

2. 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

3. 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	m_2	m_3	m_4	m_5	m_6

4. Pooling type HP

Pooling Type	Avg	Max	Glob_Avg	Glob_Max
Train-Acc	0.96	0.98	-	-
Test-Acc	0.91	0.953	-	-

5. 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

6. Observation

a. Best Model

i. Optimizer Adam
ii. LR 0.001
iii. No regularization
iv. Use Batch normalization
v. 2 CNN layers and 1 MaxPool layer

b. Training accuracy is always higher than test accuracy
c. Test accuracy is varying due to small epoch
d. Best Accuracy is 0.95

ResNet & VGG

1. Untrained

a. Hyperparameters

i. Optimizer
   1. Adam, SGD

ii. LR
   1. 0.01, 0.001

iii. Regularization
   1. Early stopping, l1, l2 (choose random Conv2D layers to apply kernel regularization)

iv. Pooling
   1. Average, max, none

v. Epochs
   1. 10, 20, 40

vi. Top-layer
   1. Adding dense (512, 1024) with dropout
   2. Adding dense (128, 512, 1024) without dropout

b. 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							

2. Pretrained HP tuning

a. Number of freezed layers

b. Epochs

c. Number of dense layers (1 or 2)

d. Neurons in dense layer (512, 512 and 256)

e. Optimizers (RMSprop, Adam, SGD)

f. Learning rate (0.001, 0.0001)

g. Regularizers (L1, L2, Dropout)

3. Best result achieved for:

a. 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

Accuracy Without freezing

b. 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

Accuracy Without freezing

Accuracy with freezing

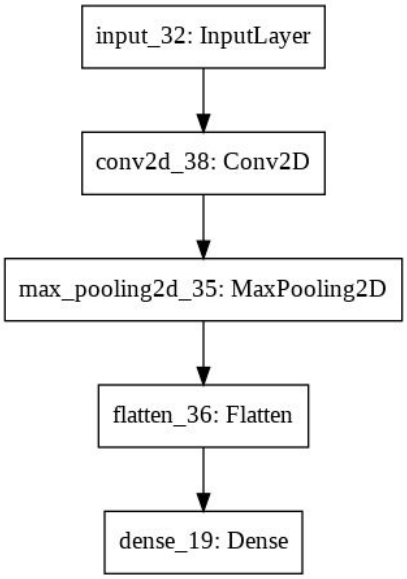
Appendix

1. CNN

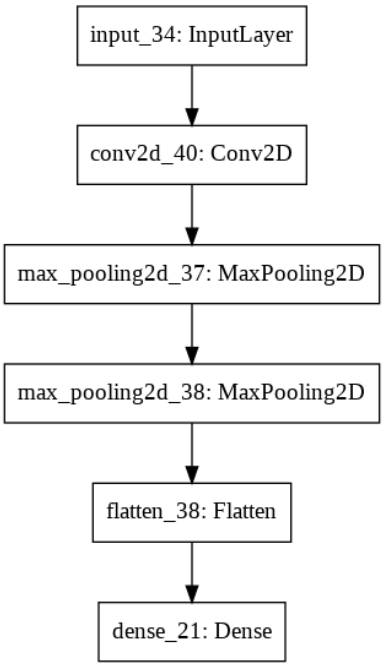
a. LR and Optimizer tuning

b. Model CNN and MP layers tuning

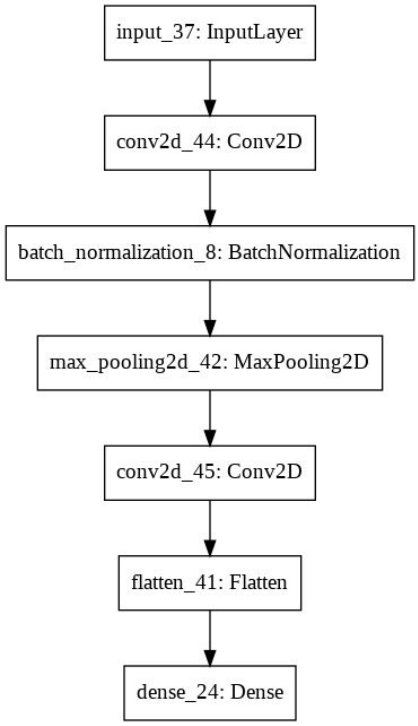
i. M_1



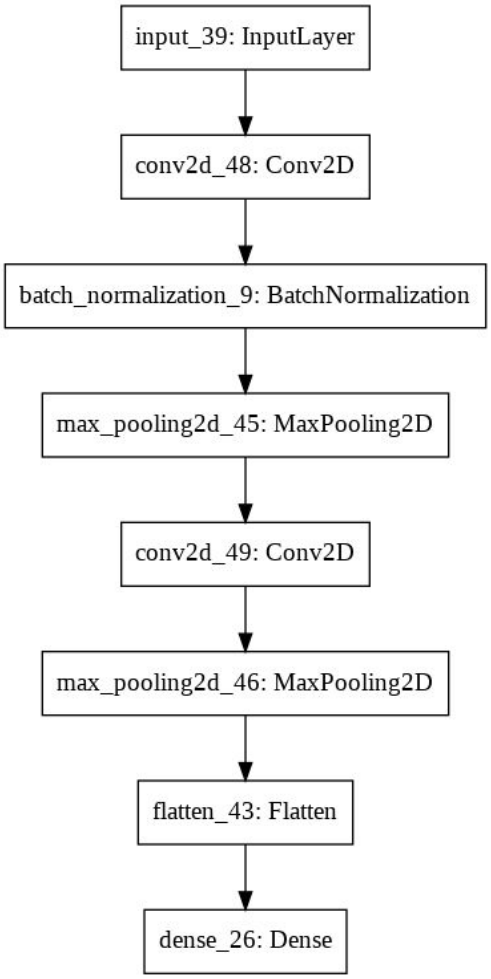
ii. M_2


input_34: InputLayer
conv2d_40: Conv2D
max_pooling2d_37: MaxPooling2D
max_pooling2d_38: MaxPooling2D
flatten_38: Flatten
dense_21: Dense

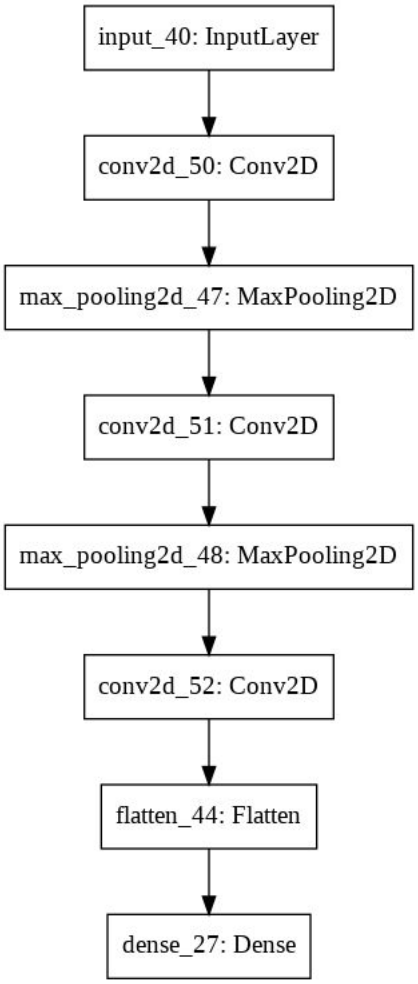
iii. M_3


input_37: InputLayer
conv2d_44: Conv2D
batch_normalization_8: BatchNormalization
max_pooling2d_42: MaxPooling2D
conv2d_45: Conv2D
flatten_41: Flatten
dense_24: Dense

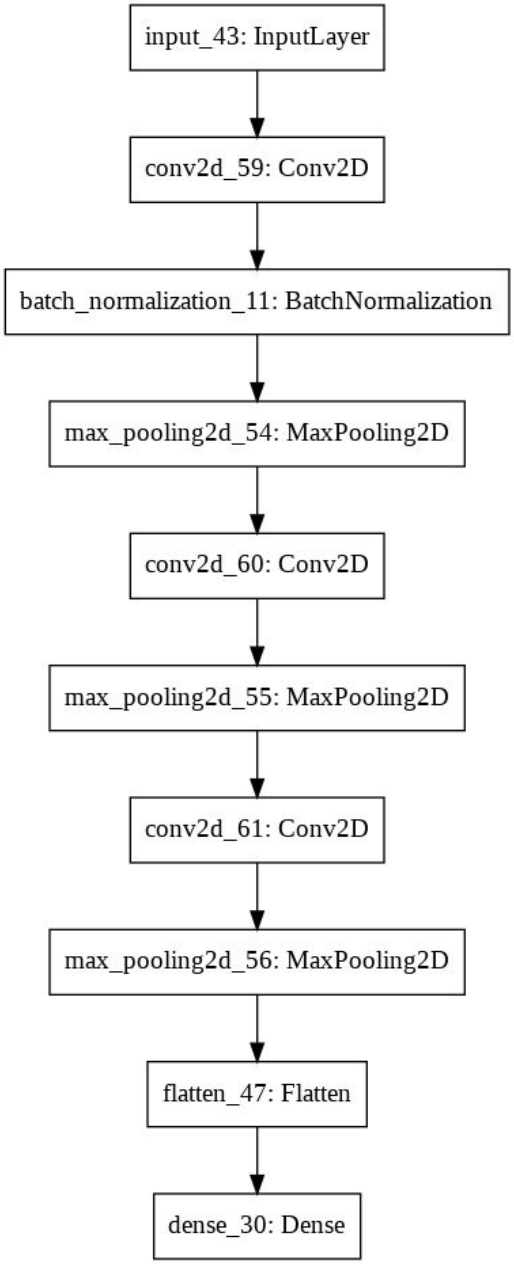
iv. M_4


input_39: InputLayer
conv2d_48: Conv2D
batch_normalization_9: BatchNormalization
max_pooling2d_45: MaxPooling2D
conv2d_49: Conv2D
max_pooling2d_46: MaxPooling2D
flatten_43: Flatten
dense_26: Dense

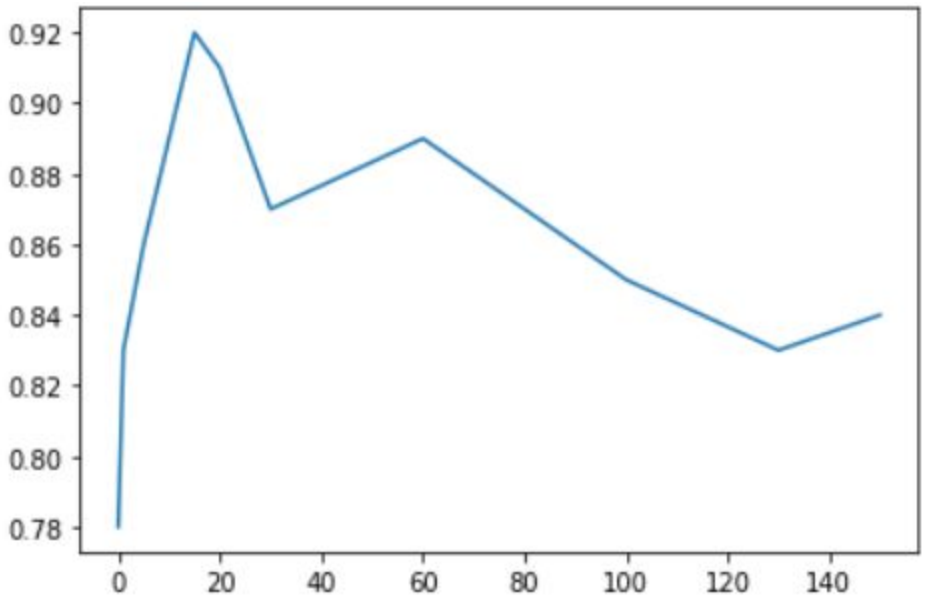
v. M_5


input_40: InputLayer
conv2d_50: Conv2D
max_pooling2d_47: MaxPooling2D
conv2d_51: Conv2D
max_pooling2d_48: MaxPooling2D
conv2d_52: Conv2D
flatten_44: Flatten
dense_27: Dense

vi. M_6

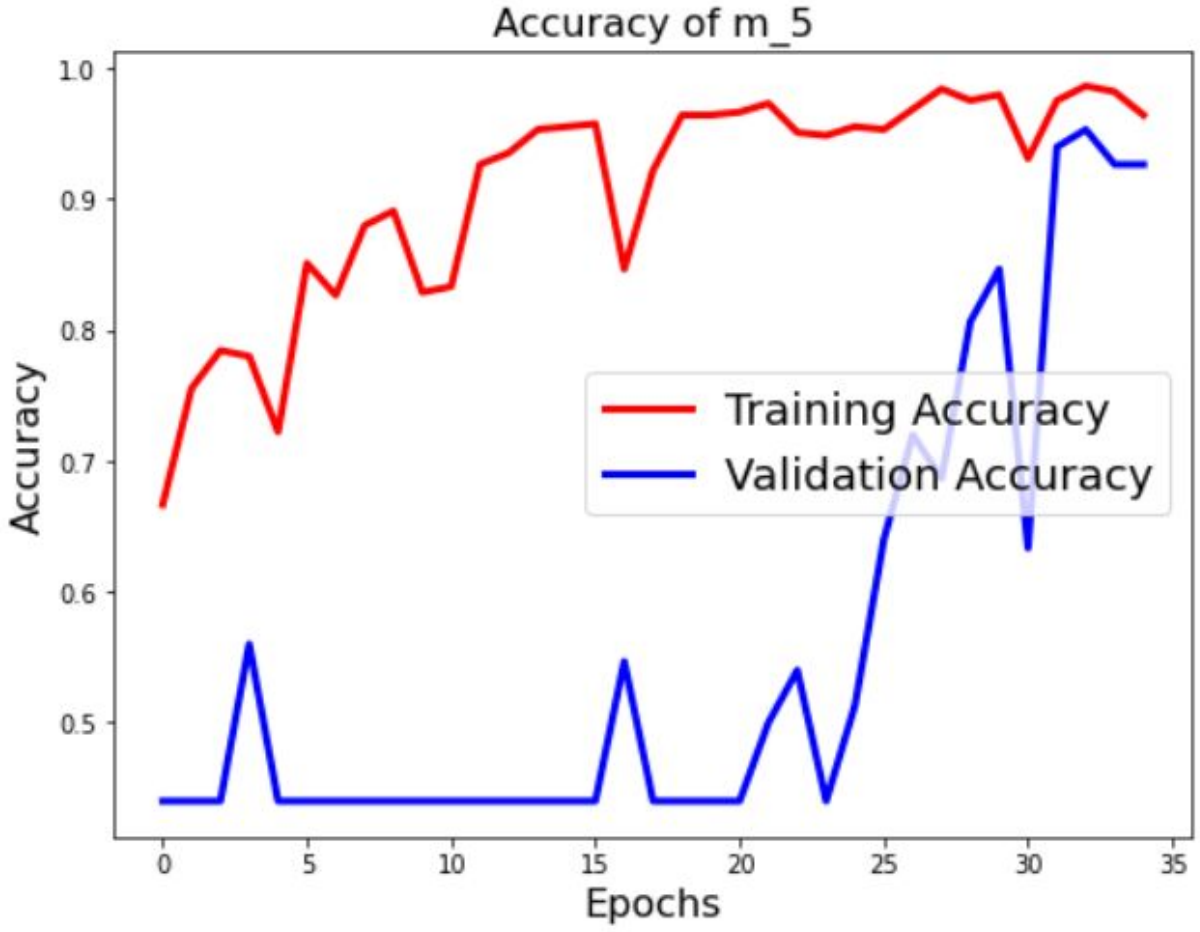

input_43: InputLayer
conv2d_59: Conv2D
batch_normalization_11: BatchNormalization
max_pooling2d_54: MaxPooling2D
conv2d_60: Conv2D
max_pooling2d_55: MaxPooling2D
conv2d_61: Conv2D
max_pooling2d_56: MaxPooling2D
flatten_47: Flatten
dense_30: Dense

c. Freezing layer plot ResNet



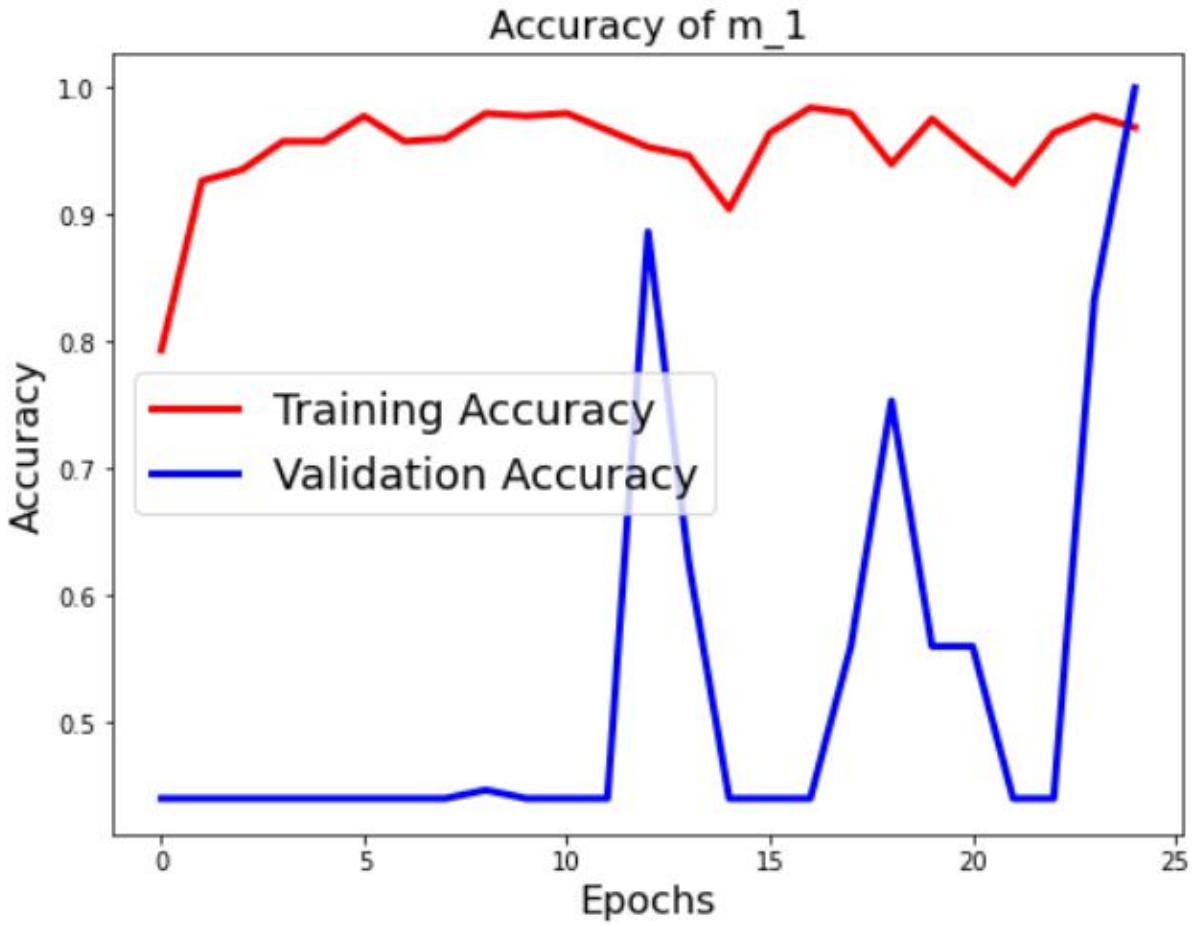
d. Without freezing Resnet

```
Loss = 0.15734563946723937
Test Accuracy = 0.94666666
```



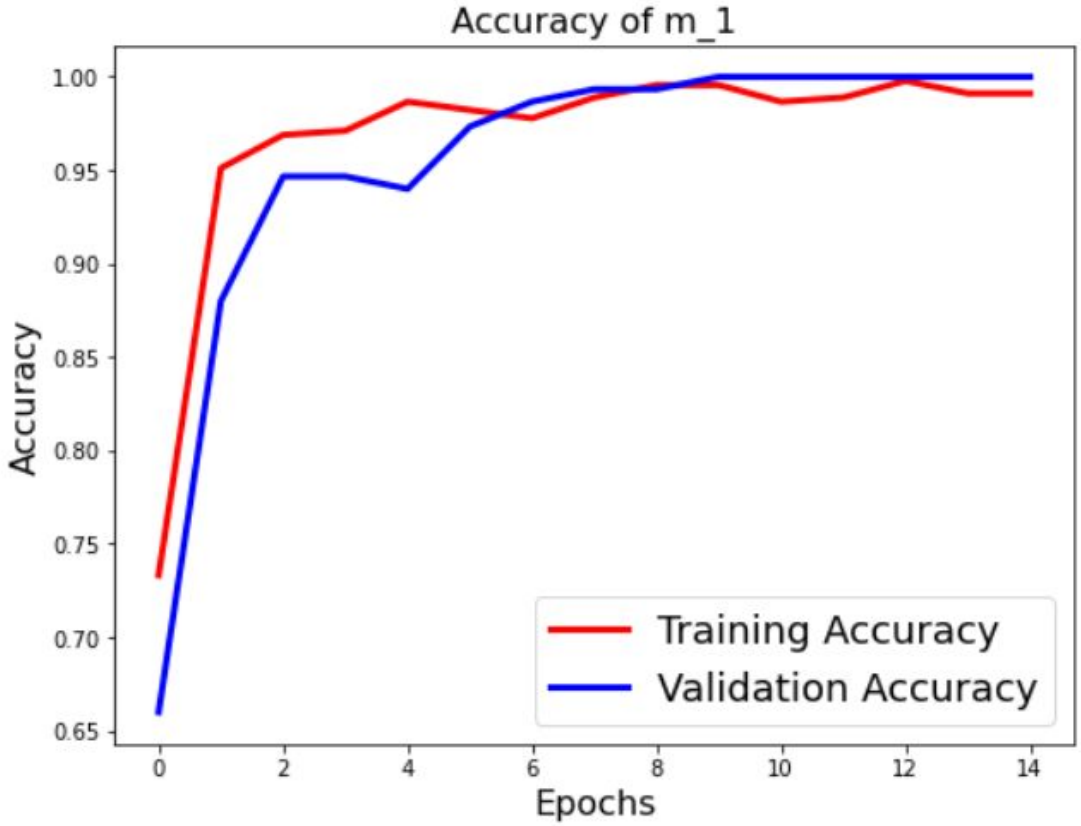
e. Without freezing VGG

```
Test Accuracy = 0.97333336
```



f. With freezing VGG

```
Loss = 0.07534479891260465
Test Accuracy = 0.98
```



g. Freezing layer plot VGG