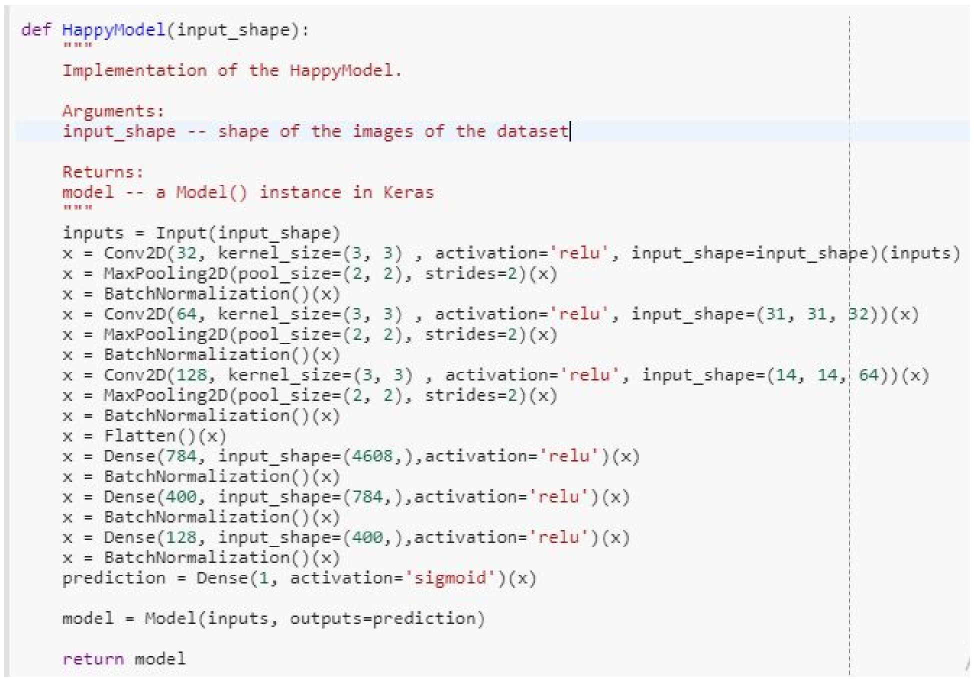
Lab 3 Deep Learning
Happy Faces

Michael Raafat Mikhail Zaki
(57)

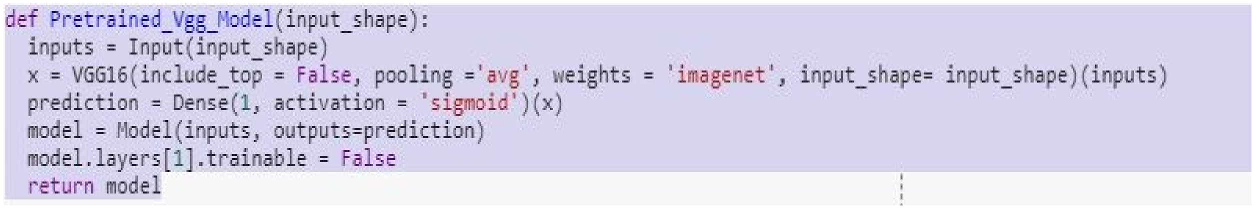
Our Model:



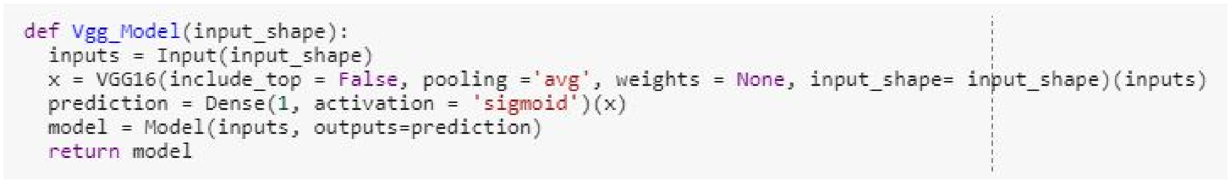
```
def HappyModel(input_shape):
    """
    Implementation of the HappyModel.

    Arguments:
    input_shape -- shape of the images of the dataset

    Returns:
    model -- a Model() instance in Keras
    """
    inputs = Input(input_shape)
    x = Conv2D(32, kernel_size=(3, 3) , activation='relu', input_shape=input_shape)(inputs)
    x = MaxPooling2D(pool_size=(2, 2), strides=2)(x)
    x = BatchNormalization()(x)
    x = Conv2D(64, kernel_size=(3, 3) , activation='relu', input_shape=(31, 31, 32))(x)
    x = MaxPooling2D(pool_size=(2, 2), strides=2)(x)
    x = BatchNormalization()(x)
    x = Conv2D(128, kernel_size=(3, 3) , activation='relu', input_shape=(14, 14, 64))(x)
    x = MaxPooling2D(pool_size=(2, 2), strides=2)(x)
    x = BatchNormalization()(x)
    x = Flatten()(x)
    x = Dense(784, input_shape=(4608,),activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dense(400, input_shape=(784,),activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dense(128, input_shape=(400,),activation='relu')(x)
    x = BatchNormalization()(x)
    prediction = Dense(1, activation='sigmoid')(x)

    model = Model(inputs, outputs=prediction)

    return model
```

- Using Adam with learning rate 0.0005 is better than SGD (testing accuracy 97.3%) as optimizer, we reached a training accuracy of 98.3% and a validation accuracy of 99.17%
- We reached a Testing accuracy of 98.6%, which is **the highest accuracy** we've reached.
- We tried Dropout as regularization but we didn't get from it the best Accuracy.
- No. of parameter is 4078305
- No. of Multiplications is 5450432
- Running Time is 270.42 Seconds

VGG:



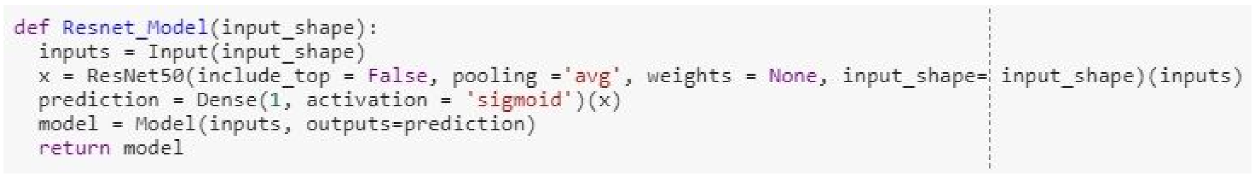
```
def Pretrained_Vgg_Model(input_shape):
  inputs = Input(input_shape)
  x = VGG16(include_top = False, pooling ='avg', weights = 'imagenet', input_shape= input_shape)(inputs)
  prediction = Dense(1, activation = 'sigmoid')(x)
  model = Model(inputs, outputs=prediction)
  model.layers[1].trainable = False
  return model
```



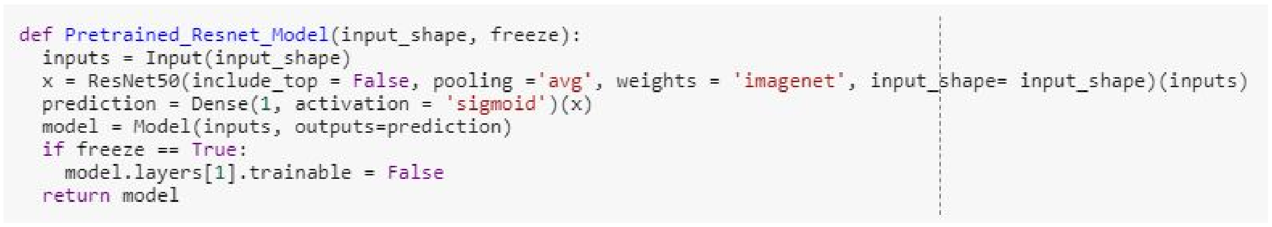
```
def Vgg_Model(input_shape):
  inputs = Input(input_shape)
  x = VGG16(include_top = False, pooling ='avg', weights = None, input_shape= input_shape)(inputs)
  prediction = Dense(1, activation = 'sigmoid')(x)
  model = Model(inputs, outputs=prediction)
  return model
```

- Using Adam is our choice we compiled it first with no pretrained (Random) weights, we reached testing accuracy of 44%.
- After Adding pretrained weights and tuning we reached 55.99%
- We tried to freeze weights of VGG and we reached our best testing accuracy with is 96%
- We put dropout as regularization and used it in tuning parameters.

ResNet



```
def Resnet_Model(input_shape):
  inputs = Input(input_shape)
  x = ResNet50(include_top = False, pooling ='avg', weights = None, input_shape= input_shape)(inputs)
  prediction = Dense(1, activation = 'sigmoid')(x)
  model = Model(inputs, outputs=prediction)
  return model
```



```
def Pretrained_Resnet_Model(input_shape, freeze):
  inputs = Input(input_shape)
  x = ResNet50(include_top = False, pooling ='avg', weights = 'imagenet', input_shape= input_shape)(inputs)
  prediction = Dense(1, activation = 'sigmoid')(x)
  model = Model(inputs, outputs=prediction)
  if freeze == True:
    model.layers[1].trainable = False
  return model
```

- Using Adam is our choice we compiled it first with no pretrained (Random) weights, we reached testing accuracy of 94.6%.
- After Adding pretrained weights and fine tuning, we reached our best testing accuracy 96.67%
- We tried to freeze weights of Resnet and performance decreased to reach 44%
- We put dropout as regularization and used it in tuning parameters.