```
import tensorflow as tf
from tensorflow import keras
import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten, Dropout, Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
```

```
!pip install tflearn
```

```
     Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
     Requirement already satisfied: tflearn in /usr/local/lib/python3.8/dist-packages (0.5.0)
     Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from tflearn) (1.15.0)
     Requirement already satisfied: Pillow in /usr/local/lib/python3.8/dist-packages (from tflearn) (7.1.2)
     Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from tflearn) (1.21.6)
```

```
# get the data

import tflearn.datasets.oxflower17 as oxflower17
x,y = oxflower17.load_data(one_hot=True)
```

```
x
```

```
           [0.6       , 0.5647059 , 0.5137255 ]]]], dtype=float32)
```

```
y.shape
```

```
    (1360, 17)
```

```
from keras.layers.normalization.batch_normalization import BatchNormalizationBase
# create a Sequential model

model = Sequential()

# 1st convolutional layer
model.add(Conv2D(filters=96,input_shape=(224,224,3), kernel_size=(11,11), strides=(4,4), padding='valid'))
model.add(Activation('relu'))

#pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
# Batch Normalization before it passing through the next layer

model.add(BatchNormalization())

#2nd convolutional layer
model.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), padding='same'))
model.add(Activation('relu'))

#pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
# Batch Normalization before it passing through the next layer

model.add(BatchNormalization())

#3rd convolutional layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(Activation('relu'))
#Batch Normalization before it passing through the next layer
model.add(BatchNormalization())

#4th convolutional layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(Activation('relu'))
#Batch Normalization
model.add(BatchNormalization())

#5th convolutional layer
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='same'))
model.add(Activation('relu'))
#pooling
model.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))

#Batch Normalization before it passing through the next layer
model.add(BatchNormalization())

# passing it to the Dense Layer
model.add(Flatten())

# 1st Dense layer

model.add(Dense(4096, input_shape=(224*224*3,)))
model.add(Activation('relu'))

# add Dropout to prevent overfitting
model.add(Dropout(0.4))
#Batch Normalization
model.add(BatchNormalization())


#2nd Dense Layer

model.add(Dense(4096))
model.add(Activation('relu'))
# add Dropout to prevent overfitting
model.add(Dropout(0.4))
#Batch Normalization
model.add(BatchNormalization())
```

```
#Output layer

model.add(Dense(17))
model.add(Activation('softmax'))

model.summary()
```

```
 2D)

 batch_normalization_13 (Bat  (None, 12, 12, 256)      1024
 chNormalization)

 conv2d_12 (Conv2D)          (None, 12, 12, 384)       885120

 activation_15 (Activation)  (None, 12, 12, 384)       0

 batch_normalization_14 (Bat  (None, 12, 12, 384)      1536
 chNormalization)

 conv2d_13 (Conv2D)          (None, 12, 12, 384)       1327488

 activation_16 (Activation)  (None, 12, 12, 384)       0

 batch_normalization_15 (Bat  (None, 12, 12, 384)      1536
 chNormalization)

 conv2d_14 (Conv2D)          (None, 12, 12, 256)       884992

 activation_17 (Activation)  (None, 12, 12, 256)       0

 max_pooling2d_8 (MaxPooling  (None, 5, 5, 256)        0
 2D)

 batch_normalization_16 (Bat  (None, 5, 5, 256)        1024
 chNormalization)

 flatten_1 (Flatten)         (None, 6400)              0

 dense_3 (Dense)             (None, 4096)              26218496

 activation_18 (Activation)  (None, 4096)              0

 dropout_2 (Dropout)         (None, 4096)              0

 batch_normalization_17 (Bat  (None, 4096)             16384
 chNormalization)

 dense_4 (Dense)             (None, 4096)              16781312

 activation_19 (Activation)  (None, 4096)              0

 dropout_3 (Dropout)         (None, 4096)              0

 batch_normalization_18 (Bat  (None, 4096)             16384
 chNormalization)

 dense_5 (Dense)             (None, 17)                69649

 activation_20 (Activation)  (None, 17)                0

=================================================================
Total params: 46,854,929
Trainable params: 46,835,793
Non-trainable params: 19,136
_________________________________________________________________
```

```
#compile
opt = tf.optimizers.Adam(learning_rate=0.01)
```

```
model.compile(loss ='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
<ipython-input-24-d8c7720726e8> in <module>
----> 1 model.compile(loss ='categorical_crossentropy', optimizer=opt, metrics=
['accuracy'])
```

7 frames

```
# train the model

model.fit(x,y,batch_size=64,epochs=5,verbose=1, validation_split=0.2,shuffle=True)
```

```
NotImplementedError: numpy() is only available when eager execution is enabled.
```

Colab paid products - Cancel contracts here

0s completed at 2:11 PM