```python
# VGG16 ON CIFAR_10

import numpy as np
from tensorflow.keras.applications.vgg16 import VGG16
import tensorflow.keras as k
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from keras.utils.np_utils import to_categorical
from tensorflow.keras import optimizers
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, 
LearningRateScheduler
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import accuracy_score


# Using VGG16 model, with weights pre-trained on ImageNet.


vgg16_model = VGG16(weights='imagenet',
                    include_top=False,
                    classes=10,
                    input_shape=(32,32,3)# input: 32x32 images with 3 
channels -> (32, 32, 3) tensors.
                   )
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 2s 0us/step
```

```python
#Define the sequential model and add th VGG's layers to it
model = Sequential()
for layer in vgg16_model.layers:
    model.add(layer)


# Adding hiddens  and output layer to our model


from tensorflow.keras.layers import Dense, Flatten, Dropout
model.add(Flatten())
model.add(Dense(512, activation='relu', name='hidden1'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu', name='hidden2'))
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax', name='predictions'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 512)               0         
_________________________________________________________________
hidden1 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
hidden2 (Dense)              (None, 256)               131328    
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0         
_________________________________________________________________
```

```
predictions (Dense)          (None, 10)                2570
=================================================================
Total params: 15,111,242
Trainable params: 15,111,242
Non-trainable params: 0
_________________________________________________________________
```

```python
#  Loading CIFAR10 data


(X_train, y_train), (X_test, y_test) = k.datasets.cifar10.load_data()

print("*****************")
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-
python.tar.gz
170500096/170498071 [==============================] - 6s 0us/step
*****************
(50000, 32, 32, 3)
(50000, 1)
(10000, 32, 32, 3)
(10000, 1)
```

```python
# Convert class vectors to binary class matrices using one hot
encoding
y_train_ohe = to_categorical(y_train, num_classes = 10)
y_test_ohe = to_categorical(y_test, num_classes = 10)

# Data normalization
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train  /= 255
X_test /= 255

print("*****************")
print(X_train.shape)
print(y_train_ohe.shape)
print(X_test.shape)
print(y_test_ohe.shape)
```

```
*****************
(50000, 32, 32, 3)
(50000, 10)
(10000, 32, 32, 3)
(10000, 10)
```

```
X_val = X_train[40000:]
y_val = y_train_ohe[40000:]
print(X_val.shape)
print(y_val.shape)
```

```
(10000, 32, 32, 3)
(10000, 10)
```

```
X_train = X_train[:40000]
y_train_ohe = y_train_ohe[:40000]
print(X_train.shape)
print(y_train_ohe.shape)
```

```
(40000, 32, 32, 3)
(40000, 10)
```

```
# TRAINING THE CNN ON THE TRAIN/VALIDATION DATA
```

```
# initiate SGD optimizer
sgd = optimizers.SGD(lr=0.001, momentum=0.9)

# For a multi-class classification problem
model.compile(loss='categorical_crossentropy',optimizer=
sgd,metrics=['accuracy'])


def lr_scheduler(epoch):
    return 0.001 * (0.5 ** (epoch // 20))
reduce_lr = LearningRateScheduler(lr_scheduler)

mc = ModelCheckpoint('./weights.h5', monitor='val_accuracy',
save_best_only=True, mode='max')


# initialize the number of epochs and batch size
EPOCHS = 100
BS = 128

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")
```

```python
# train the model
history = model.fit_generator(
    aug.flow(X_train,y_train_ohe, batch_size=BS),
    validation_data=(X_val,y_val),
    steps_per_epoch=len(X_train) // BS,
    epochs=EPOCHS,
    callbacks=[reduce_lr,mc])

#We load the best weights saved by the ModelCheckpoint
model.load_weights('./weights.h5')
```

```
Epoch 1/100
312/312 [==============================] - 28s 91ms/step - loss:
1.7427 - accuracy: 0.3695 - val_loss: 1.2011 - val_accuracy: 0.5747
Epoch 2/100
312/312 [==============================] - 28s 89ms/step - loss:
1.1861 - accuracy: 0.5932 - val_loss: 0.8432 - val_accuracy: 0.7064
Epoch 3/100
312/312 [==============================] - 27s 87ms/step - loss:
1.0126 - accuracy: 0.6566 - val_loss: 0.8207 - val_accuracy: 0.7203
Epoch 4/100
312/312 [==============================] - 27s 87ms/step - loss:
0.9008 - accuracy: 0.6966 - val_loss: 0.6831 - val_accuracy: 0.7663
Epoch 5/100
312/312 [==============================] - 27s 86ms/step - loss:
0.8286 - accuracy: 0.7221 - val_loss: 0.7118 - val_accuracy: 0.7601
Epoch 6/100
312/312 [==============================] - 27s 86ms/step - loss:
0.7733 - accuracy: 0.7436 - val_loss: 0.6239 - val_accuracy: 0.7872
Epoch 7/100
312/312 [==============================] - 26s 84ms/step - loss:
0.7345 - accuracy: 0.7552 - val_loss: 0.5286 - val_accuracy: 0.8161
Epoch 8/100
312/312 [==============================] - 27s 86ms/step - loss:
0.6861 - accuracy: 0.7705 - val_loss: 0.5708 - val_accuracy: 0.8028
Epoch 9/100
312/312 [==============================] - 27s 87ms/step - loss:
0.6710 - accuracy: 0.7766 - val_loss: 0.5145 - val_accuracy: 0.8198
Epoch 10/100
312/312 [==============================] - 27s 86ms/step - loss:
0.6391 - accuracy: 0.7870 - val_loss: 0.5257 - val_accuracy: 0.8194
Epoch 11/100
312/312 [==============================] - 27s 87ms/step - loss:
0.6121 - accuracy: 0.7956 - val_loss: 0.4811 - val_accuracy: 0.8367
Epoch 12/100
312/312 [==============================] - 27s 86ms/step - loss:
0.5897 - accuracy: 0.8011 - val_loss: 0.4771 - val_accuracy: 0.8383
Epoch 13/100
312/312 [==============================] - 27s 85ms/step - loss:
0.5740 - accuracy: 0.8096 - val_loss: 0.4707 - val_accuracy: 0.8406
```

```
Epoch 14/100
312/312 [==============================] - 26s 84ms/step - loss:
0.5464 - accuracy: 0.8174 - val_loss: 0.4596 - val_accuracy: 0.8437
Epoch 15/100
312/312 [==============================] - 27s 87ms/step - loss:
0.5357 - accuracy: 0.8210 - val_loss: 0.4389 - val_accuracy: 0.8500
Epoch 16/100
312/312 [==============================] - 27s 88ms/step - loss:
0.5186 - accuracy: 0.8284 - val_loss: 0.5113 - val_accuracy: 0.8346
Epoch 17/100
312/312 [==============================] - 27s 87ms/step - loss:
0.5084 - accuracy: 0.8302 - val_loss: 0.4470 - val_accuracy: 0.8503
Epoch 18/100
312/312 [==============================] - 27s 86ms/step - loss:
0.4986 - accuracy: 0.8331 - val_loss: 0.4430 - val_accuracy: 0.8491
Epoch 19/100
312/312 [==============================] - 27s 86ms/step - loss:
0.4874 - accuracy: 0.8375 - val_loss: 0.4756 - val_accuracy: 0.8404
Epoch 20/100
312/312 [==============================] - 27s 86ms/step - loss:
0.4699 - accuracy: 0.8438 - val_loss: 0.4029 - val_accuracy: 0.8615
Epoch 21/100
312/312 [==============================] - 27s 87ms/step - loss:
0.4269 - accuracy: 0.8570 - val_loss: 0.3865 - val_accuracy: 0.8718
Epoch 22/100
312/312 [==============================] - 27s 86ms/step - loss:
0.4122 - accuracy: 0.8637 - val_loss: 0.4152 - val_accuracy: 0.8615
Epoch 23/100
312/312 [==============================] - 27s 87ms/step - loss:
0.4075 - accuracy: 0.8636 - val_loss: 0.3865 - val_accuracy: 0.8712
Epoch 24/100
312/312 [==============================] - 27s 86ms/step - loss:
0.4017 - accuracy: 0.8652 - val_loss: 0.4007 - val_accuracy: 0.8680
Epoch 25/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3936 - accuracy: 0.8683 - val_loss: 0.4052 - val_accuracy: 0.8678
Epoch 26/100
312/312 [==============================] - 27s 85ms/step - loss:
0.3886 - accuracy: 0.8715 - val_loss: 0.4061 - val_accuracy: 0.8657
Epoch 27/100
312/312 [==============================] - 26s 84ms/step - loss:
0.3811 - accuracy: 0.8721 - val_loss: 0.4433 - val_accuracy: 0.8568
Epoch 28/100
312/312 [==============================] - 27s 88ms/step - loss:
0.3790 - accuracy: 0.8733 - val_loss: 0.3856 - val_accuracy: 0.8731
Epoch 29/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3693 - accuracy: 0.8751 - val_loss: 0.4282 - val_accuracy: 0.8636
Epoch 30/100
312/312 [==============================] - 27s 86ms/step - loss:
```

```
0.3630 - accuracy: 0.8785 - val_loss: 0.3975 - val_accuracy: 0.8683
Epoch 31/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3585 - accuracy: 0.8809 - val_loss: 0.3916 - val_accuracy: 0.8687
Epoch 32/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3563 - accuracy: 0.8800 - val_loss: 0.3860 - val_accuracy: 0.8711
Epoch 33/100
312/312 [==============================] - 26s 85ms/step - loss:
0.3531 - accuracy: 0.8827 - val_loss: 0.4171 - val_accuracy: 0.8629
Epoch 34/100
312/312 [==============================] - 27s 85ms/step - loss:
0.3435 - accuracy: 0.8847 - val_loss: 0.4028 - val_accuracy: 0.8715
Epoch 35/100
312/312 [==============================] - 27s 85ms/step - loss:
0.3397 - accuracy: 0.8857 - val_loss: 0.3818 - val_accuracy: 0.8750
Epoch 36/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3349 - accuracy: 0.8875 - val_loss: 0.3664 - val_accuracy: 0.8824
Epoch 37/100
312/312 [==============================] - 27s 86ms/step - loss:
0.3300 - accuracy: 0.8875 - val_loss: 0.3722 - val_accuracy: 0.8786
Epoch 38/100
312/312 [==============================] - 27s 85ms/step - loss:
0.3297 - accuracy: 0.8893 - val_loss: 0.4055 - val_accuracy: 0.8732
Epoch 39/100
312/312 [==============================] - 26s 84ms/step - loss:
0.3234 - accuracy: 0.8897 - val_loss: 0.3926 - val_accuracy: 0.8736
Epoch 40/100
312/312 [==============================] - 26s 85ms/step - loss:
0.3171 - accuracy: 0.8953 - val_loss: 0.3752 - val_accuracy: 0.8809
Epoch 41/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2916 - accuracy: 0.9020 - val_loss: 0.3789 - val_accuracy: 0.8778
Epoch 42/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2858 - accuracy: 0.9043 - val_loss: 0.3727 - val_accuracy: 0.8815
Epoch 43/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2861 - accuracy: 0.9033 - val_loss: 0.3989 - val_accuracy: 0.8741
Epoch 44/100
312/312 [==============================] - 27s 88ms/step - loss:
0.2778 - accuracy: 0.9061 - val_loss: 0.3753 - val_accuracy: 0.8844
Epoch 45/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2795 - accuracy: 0.9065 - val_loss: 0.3781 - val_accuracy: 0.8807
Epoch 46/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2773 - accuracy: 0.9087 - val_loss: 0.3798 - val_accuracy: 0.8821
Epoch 47/100
```

```
312/312 [==============================] - 27s 86ms/step - loss:
0.2757 - accuracy: 0.9074 - val_loss: 0.3759 - val_accuracy: 0.8807
Epoch 48/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2693 - accuracy: 0.9098 - val_loss: 0.3900 - val_accuracy: 0.8771
Epoch 49/100
312/312 [==============================] - 27s 87ms/step - loss:
0.2680 - accuracy: 0.9122 - val_loss: 0.3671 - val_accuracy: 0.8860
Epoch 50/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2629 - accuracy: 0.9108 - val_loss: 0.3889 - val_accuracy: 0.8791
Epoch 51/100
312/312 [==============================] - 26s 85ms/step - loss:
0.2643 - accuracy: 0.9104 - val_loss: 0.3740 - val_accuracy: 0.8809
Epoch 52/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2618 - accuracy: 0.9121 - val_loss: 0.3886 - val_accuracy: 0.8790
Epoch 53/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2580 - accuracy: 0.9139 - val_loss: 0.3877 - val_accuracy: 0.8797
Epoch 54/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2599 - accuracy: 0.9143 - val_loss: 0.3687 - val_accuracy: 0.8836
Epoch 55/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2516 - accuracy: 0.9149 - val_loss: 0.3756 - val_accuracy: 0.8835
Epoch 56/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2498 - accuracy: 0.9144 - val_loss: 0.3834 - val_accuracy: 0.8815
Epoch 57/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2517 - accuracy: 0.9161 - val_loss: 0.3720 - val_accuracy: 0.8841
Epoch 58/100
312/312 [==============================] - 26s 82ms/step - loss:
0.2499 - accuracy: 0.9161 - val_loss: 0.3821 - val_accuracy: 0.8829
Epoch 59/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2459 - accuracy: 0.9165 - val_loss: 0.3699 - val_accuracy: 0.8848
Epoch 60/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2439 - accuracy: 0.9184 - val_loss: 0.3752 - val_accuracy: 0.8862
Epoch 61/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2371 - accuracy: 0.9209 - val_loss: 0.3754 - val_accuracy: 0.8861
Epoch 62/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2294 - accuracy: 0.9228 - val_loss: 0.3691 - val_accuracy: 0.8864
Epoch 63/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2245 - accuracy: 0.9245 - val_loss: 0.4016 - val_accuracy: 0.8813
```

```
Epoch 64/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2248 - accuracy: 0.9252 - val_loss: 0.3694 - val_accuracy: 0.8855
Epoch 65/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2235 - accuracy: 0.9246 - val_loss: 0.3744 - val_accuracy: 0.8859
Epoch 66/100
312/312 [==============================] - 26s 85ms/step - loss:
0.2253 - accuracy: 0.9237 - val_loss: 0.3948 - val_accuracy: 0.8811
Epoch 67/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2198 - accuracy: 0.9272 - val_loss: 0.3953 - val_accuracy: 0.8810
Epoch 68/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2203 - accuracy: 0.9276 - val_loss: 0.3717 - val_accuracy: 0.8855
Epoch 69/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2183 - accuracy: 0.9269 - val_loss: 0.3754 - val_accuracy: 0.8857
Epoch 70/100
312/312 [==============================] - 26s 82ms/step - loss:
0.2170 - accuracy: 0.9270 - val_loss: 0.3817 - val_accuracy: 0.8857
Epoch 71/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2123 - accuracy: 0.9290 - val_loss: 0.3755 - val_accuracy: 0.8855
Epoch 72/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2116 - accuracy: 0.9284 - val_loss: 0.3871 - val_accuracy: 0.8833
Epoch 73/100
312/312 [==============================] - 26s 84ms/step - loss:
0.2118 - accuracy: 0.9278 - val_loss: 0.3785 - val_accuracy: 0.8863
Epoch 74/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2073 - accuracy: 0.9302 - val_loss: 0.3857 - val_accuracy: 0.8827
Epoch 75/100
312/312 [==============================] - 25s 81ms/step - loss:
0.2120 - accuracy: 0.9290 - val_loss: 0.3819 - val_accuracy: 0.8844
Epoch 76/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2081 - accuracy: 0.9304 - val_loss: 0.3968 - val_accuracy: 0.8830
Epoch 77/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2117 - accuracy: 0.9285 - val_loss: 0.3976 - val_accuracy: 0.8811
Epoch 78/100
312/312 [==============================] - 26s 83ms/step - loss:
0.2039 - accuracy: 0.9325 - val_loss: 0.3883 - val_accuracy: 0.8855
Epoch 79/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2079 - accuracy: 0.9295 - val_loss: 0.3897 - val_accuracy: 0.8840
Epoch 80/100
312/312 [==============================] - 26s 84ms/step - loss:
```

```
0.2046 - accuracy: 0.9317 - val_loss: 0.3822 - val_accuracy: 0.8837
Epoch 81/100
312/312 [==============================] - 26s 82ms/step - loss:
0.2014 - accuracy: 0.9327 - val_loss: 0.3898 - val_accuracy: 0.8832
Epoch 82/100
312/312 [==============================] - 27s 85ms/step - loss:
0.2004 - accuracy: 0.9321 - val_loss: 0.3879 - val_accuracy: 0.8845
Epoch 83/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2011 - accuracy: 0.9332 - val_loss: 0.3805 - val_accuracy: 0.8854
Epoch 84/100
312/312 [==============================] - 27s 86ms/step - loss:
0.2035 - accuracy: 0.9317 - val_loss: 0.3775 - val_accuracy: 0.8846
Epoch 85/100
312/312 [==============================] - 27s 87ms/step - loss:
0.1959 - accuracy: 0.9348 - val_loss: 0.3889 - val_accuracy: 0.8839
Epoch 86/100
312/312 [==============================] - 27s 87ms/step - loss:
0.1942 - accuracy: 0.9341 - val_loss: 0.3723 - val_accuracy: 0.8870
Epoch 87/100
312/312 [==============================] - 26s 83ms/step - loss:
0.1947 - accuracy: 0.9353 - val_loss: 0.3821 - val_accuracy: 0.8861
Epoch 88/100
312/312 [==============================] - 26s 83ms/step - loss:
0.1923 - accuracy: 0.9347 - val_loss: 0.3891 - val_accuracy: 0.8841
Epoch 89/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1902 - accuracy: 0.9351 - val_loss: 0.3847 - val_accuracy: 0.8861
Epoch 90/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1944 - accuracy: 0.9342 - val_loss: 0.3820 - val_accuracy: 0.8860
Epoch 91/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1916 - accuracy: 0.9355 - val_loss: 0.3867 - val_accuracy: 0.8856
Epoch 92/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1894 - accuracy: 0.9357 - val_loss: 0.3894 - val_accuracy: 0.8873
Epoch 93/100
312/312 [==============================] - 25s 81ms/step - loss:
0.1884 - accuracy: 0.9370 - val_loss: 0.3957 - val_accuracy: 0.8845
Epoch 94/100
312/312 [==============================] - 27s 86ms/step - loss:
0.1917 - accuracy: 0.9354 - val_loss: 0.3807 - val_accuracy: 0.8875
Epoch 95/100
312/312 [==============================] - 27s 85ms/step - loss:
0.1873 - accuracy: 0.9360 - val_loss: 0.3879 - val_accuracy: 0.8852
Epoch 96/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1816 - accuracy: 0.9396 - val_loss: 0.3859 - val_accuracy: 0.8864
Epoch 97/100
```

```
312/312 [==============================] - 26s 84ms/step - loss:
0.1889 - accuracy: 0.9357 - val_loss: 0.3839 - val_accuracy: 0.8858
Epoch 98/100
312/312 [==============================] - 26s 84ms/step - loss:
0.1855 - accuracy: 0.9372 - val_loss: 0.3915 - val_accuracy: 0.8861
Epoch 99/100
312/312 [==============================] - 26s 83ms/step - loss:
0.1912 - accuracy: 0.9354 - val_loss: 0.3869 - val_accuracy: 0.8860
Epoch 100/100
312/312 [==============================] - 26s 83ms/step - loss:
0.1902 - accuracy: 0.9359 - val_loss: 0.3874 - val_accuracy: 0.8856
```

```
train_loss, train_accuracy =
model.evaluate_generator(aug.flow(X_train,y_train_ohe, batch_size=BS),
156)
print('Training loss: {}\nTraining accuracy: {}'.format(train_loss,
train_accuracy))
```

```
Training loss: 0.1625007838010788
Training accuracy: 0.9458132982254028
```

```
val_loss, val_accuracy = model.evaluate(X_val, y_val)
print('Validation loss: {}\nValidation accuracy: {}'.format(val_loss,
val_accuracy))
```

```
313/313 [==============================] - 2s 7ms/step - loss: 0.3807
- accuracy: 0.8875
Validation loss: 0.3807496428489685
Validation accuracy: 0.887499988079071
```

```
test_loss, test_accuracy = model.evaluate(X_test,y_test_ohe,)
print('Testing loss: {}\nTesting accuracy: {}'.format(test_loss,
test_accuracy))
```

```
313/313 [==============================] - 2s 7ms/step - loss: 0.4064
- accuracy: 0.8825
Testing loss: 0.4063931703567505
Testing accuracy: 0.8824999928474426
```