InceptionV3: Experiment - 2

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client

Enter your authorization code:
··········
Mounted at /content/gdrive
```

```
train_data_dir = '/content/gdrive/My Drive/ML/project-3/flowers'
img_width, img_height = 299, 299
batch_size = 64
EPOCHS = 10
```

```
from keras.preprocessing.image import ImageDataGenerator
image_datagen = ImageDataGenerator(
    rescale=1./255,
    vertical_flip = True,
    horizontal_flip = True,
    rotation_range=20,
    shear_range=0.05,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    validation_split=0.2,
    channel_shift_range=0.1
)

train_gen = image_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode="categorical",
        subset="training")

valid_gen = image_datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_height, img_width),
        batch_size=batch_size,
        class_mode="categorical",
        subset="validation")
```

```
Using TensorFlow backend.
Found 2131 images belonging to 5 classes.
Found 531 images belonging to 5 classes.
```

```
from keras.callbacks import EarlyStopping, TensorBoard, CSVLogger, ReduceLROnPlat
```

```
# Callbacks - Get some information while the model is training.

# Stop training when a monitored metric has stopped improving.
earlystop = EarlyStopping(
    monitor='val_loss',
    min_delta=0.001,
    patience=10,
    verbose=1,
    mode='auto'
)

# Callback that streams epoch results to a csv file.
csvlogger = CSVLogger(
    filename= "training_csv.log",
    separator = ",",
    append = False
)

# Reduce learning rate when a metric has stopped improving.
reduce = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.1,
    patience=3,
    verbose=1,
    mode='auto',
)
```

Hyperparameters

```
from keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(img_
print('Loaded model!')
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releas
87916544/87910968 [==============================] - 1s 0us/step
Loaded model!
```

Freeze the layers in base_model

```
for layer in base_model.layers:
    layer.trainable = False
```

HYPER-PARAMETERS

```
second_dense_512 = [0, 1]
dropout = [0, 1]
```

```
import time
from keras.layers import Dense,Flatten,Dropout,Concatenate,GlobalAveragePooling2
from keras.models import Sequential,Input,Model

for dense2 in second_dense_512:
    for drop in dropout:

        NAME = "flowers-inception-dense{}-drop{}-{}".format(dense2, drop, int(ti
        print(NAME)
        logdir = "logs/flowers-inception/{}/".format(NAME)

        # ModelCheckpoint - Callback to save the Keras model or model weights at
        checkpoint = ModelCheckpoint(
            '{}base.model'.format(logdir),
            monitor='val_loss',
            mode='min',
            save_weights_only=True,
            save_best_only = True,
            verbose = 1)

        # TensorBoard provides the visualization and tooling needed for machine
        tensorboard = TensorBoard(
            log_dir = logdir,
            histogram_freq=0,
            batch_size=batch_size,
            write_graph=True,
            write_grads=True,
            write_images=False,
        )

        x = base_model.output
        x = GlobalAveragePooling2D()(x)
        x = Dense(1024)(x)
        x = BatchNormalization()(x)
        x = Activation("relu")(x)
        if drop == 1 : x = Dropout(0.3)(x)
        if dense2 == 1 :
            x = Dense(512)(x)
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            if drop == 1 : x = Dropout(0.3)(x)

        predictions = Dense(5, activation='softmax')(x)

        model = Model(base_model.input, predictions)


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

```
flowers-inception-dense0-drop0-1594905935
flowers-inception-dense0-drop1-1594905935
/usr/local/lib/python3.6/dist-packages/keras/callbacks/tensorboard_v2.py:92:
  warnings.warn('The TensorBoard callback `batch_size` argument '
/usr/local/lib/python3.6/dist-packages/keras/callbacks/tensorboard_v2.py:97:
  warnings.warn('The TensorBoard callback does not support '
```

```
import pandas as pd
pd.set_option('max_colwidth', -1)
layers = [(layer, layer.name, layer.trainable) for layer in model.layers]
print(pd.DataFrame(layers, columns=['Layer Type', 'Layer Name', 'Layer Trainable'
```

```
                                                                    Layer Type
0    <keras.engine.input_layer.InputLayer object at 0x7f95b4f5d908>
1    <keras.layers.convolutional.Conv2D object at 0x7f9572c11a20>
2    <keras.layers.normalization.BatchNormalization object at 0x7f9572c11b00>
3    <keras.layers.core.Activation object at 0x7f9572c11908>
4    <keras.layers.convolutional.Conv2D object at 0x7f9572c11fd0>
..                                                                 ...
316  <keras.layers.core.Dense object at 0x7f956af3f4a8>
317  <keras.layers.normalization.BatchNormalization object at 0x7f956af53160>
318  <keras.layers.core.Activation object at 0x7f956af68e80>
319  <keras.layers.core.Dropout object at 0x7f956aef7a58>
320  <keras.layers.core.Dense object at 0x7f956aef73c8>

[321 rows x 3 columns]
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: FutureWarning
```

```
history = model.fit_generator(
            train_gen,
            steps_per_epoch = train_gen.n // train_gen.batch_size, # normalde len
            epochs= EPOCHS,
            validation_data = valid_gen,
            validation_steps=valid_gen.n // valid_gen.batch_size, # normalde len(
            verbose=1,
            callbacks=[checkpoint,tensorboard,csvlogger,reduce,earlystop])
```

```
Epoch 1/10
33/33 [==============================] - 758s 23s/step - loss: 0.4814 - accur

Epoch 00001: val_loss improved from inf to 0.90934, saving model to logs/flow
Epoch 2/10
33/33 [==============================] - 664s 20s/step - loss: 0.2185 - accur

Epoch 00002: val_loss improved from 0.90934 to 0.48345, saving model to logs/
Epoch 3/10
33/33 [==============================] - 657s 20s/step - loss: 0.1814 - accur

Epoch 00003: val_loss did not improve from 0.48345
Epoch 4/10
33/33 [==============================] - 669s 20s/step - loss: 0.1648 - accur

Epoch 00004: val_loss did not improve from 0.48345
Epoch 5/10
33/33 [==============================] - 673s 20s/step - loss: 0.1356 - accur

Epoch 00005: val_loss did not improve from 0.48345

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.00010000000474974
Epoch 6/10
33/33 [==============================] - 649s 20s/step - loss: 0.1137 - accur

Epoch 00006: val_loss did not improve from 0.48345
Epoch 7/10
33/33 [==============================] - 659s 20s/step - loss: 0.1146 - accur
```

Save the model

```
# model.save('flower_inceptionV3.h5')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent
call last)
<ipython-input-1-d0514e483947> in <module>()
----> 1 model.save('flower_inceptionV3.h5')

NameError: name 'model' is not defined
```

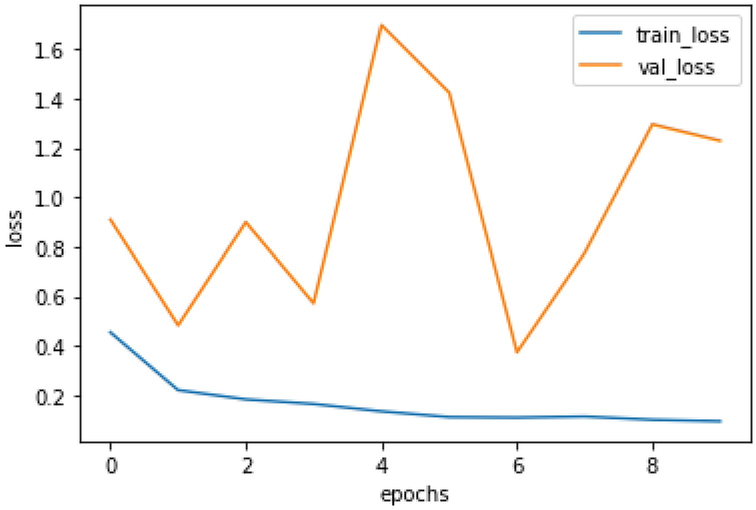
Learning plots

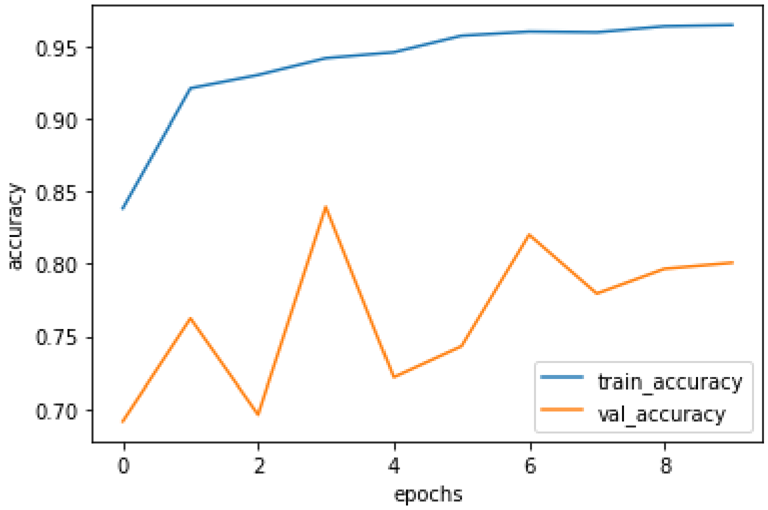
```
import matplotlib.pyplot as plt

# plot loss and accuracy image
history_dict = history.history
train_loss = history_dict["loss"]
train_accuracy = history_dict["accuracy"]
val_loss = history_dict["val_loss"]
val_accuracy = history_dict["val_accuracy"]
```

```
# figure 1
plt.figure()
plt.plot(range(EPOCHS), train_loss, label='train_loss')
plt.plot(range(EPOCHS), val_loss, label='val_loss')
plt.legend()
plt.xlabel('epochs')
plt.ylabel('loss')

# figure 2
plt.figure()
plt.plot(range(EPOCHS), train_accuracy, label='train_accuracy')
plt.plot(range(EPOCHS), val_accuracy, label='val_accuracy')
plt.legend()
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.show()
```





References

https://www.kaggle.com/emirhanozkan/flower-classification-inceptionv3

https://www.kaggle.com/yaoyi970403/flowers-rrecognition-project-acc-96-6

https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl

https://www.kaggle.com/rajmehra03/flower-recognition-cnn-keras