catsVdogs_DataAug

September 3, 2021

1 Cat vs Dogs classifier w/ Data Augmentation

1.1 Importing required modules

[1]:
```
from __future__ import absolute_import, division, print_function

import os
import matplotlib.pyplot as plt
import numpy as np

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

from jupyterthemes import jtplot
jtplot.style(theme='onedork', figsize=(16,9))

tf.config.list_physical_devices('GPU')
```

[1]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

1.2 Importing Data

[2]:
```
_URL = 'https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.
↪zip'
zip_dir = tf.keras.utils.get_file('cats_and_dogs_filtered.zip', origin=_URL,␣
↪extract=True)
```

[3]:
```
zip_dir_base = os.path.dirname(zip_dir)
print(zip_dir_base)
!tree $zip_dir_base
```

```
C:\Users\Arunabh\.keras\datasets
Folder PATH listing for volume Windows
Volume serial number is 000000F2 929F:B39D
C:\USERS\ARUNABH\.KERAS\DATASETS
  cats_and_dogs_filtered
      train
          cats
```

```
        dogs
    validation
        cats
        dogs
```

[4]:

```
base_dir = os.path.join(os.path.dirname(zip_dir), 'cats_and_dogs_filtered')
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

train_cats_dir = os.path.join(train_dir, 'cats')
train_dogs_dir = os.path.join(train_dir, 'dogs')
validation_cats_dir = os.path.join(validation_dir, 'cats')
validation_dogs_dir = os.path.join(validation_dir, 'dogs')
```

1.3 Analysing the data

[5]:

```
num_cats_tr = len(os.listdir(train_cats_dir))
num_dogs_tr = len(os.listdir(train_dogs_dir))

num_cats_val = len(os.listdir(validation_cats_dir))
num_dogs_val = len(os.listdir(validation_dogs_dir))

total_train = num_cats_tr + num_dogs_tr
total_val = num_cats_val + num_dogs_val

print('total training cat images    :', num_cats_tr)
print('total training dog images    :', num_dogs_tr)

print('total validation cat images  :', num_cats_val)
print('total validation dog images  :', num_dogs_val)
print('--')
print('total training images        :', total_train)
print('total validation images      :', total_val)
```

```
total training cat images    : 1000
total training dog images    : 1000
total validation cat images  : 500
total validation dog images  : 500
--
total training images        : 2000
total validation images      : 1000
```

1.4 Data preparation

[6]:

```
BATCH_SIZE = 100
IMG_SHAPE = 150
```

1.4.1 Data Augmentation

[12]:
```
def plotImages(images_arr):

    """Plot images in a 1x5 grid"""

    fig, axes = plt.subplots(1, 5, figsize=(20, 20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
        ax.grid(False)
    plt.tight_layout()
    plt.show()
```

Horizontal flip

[8]:
```
image_gen = ImageDataGenerator(rescale=1./255, horizontal_flip=True)
train_data_gen = image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                                directory=train_dir,
                                                shuffle=True,
                                                target_size=(IMG_SHAPE,␣
 ↪IMG_SHAPE))
```

```
Found 2000 images belonging to 2 classes.
```

[13]:
```
augmented_images = [train_data_gen[0][0][0] for i in range(5)]
plotImages(augmented_images)
```

Rotation

[14]:
```
image_gen = ImageDataGenerator(rescale=1./255, rotation_range=45)

train_data_gen = image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                                directory=train_dir,
                                                shuffle=True,
                                                target_size=(IMG_SHAPE,␣
 ↪IMG_SHAPE))
```

```
Found 2000 images belonging to 2 classes.
```

```
[15]: augmented_images = [train_data_gen[0][0][0] for i in range(5)]
      plotImages(augmented_images)
```

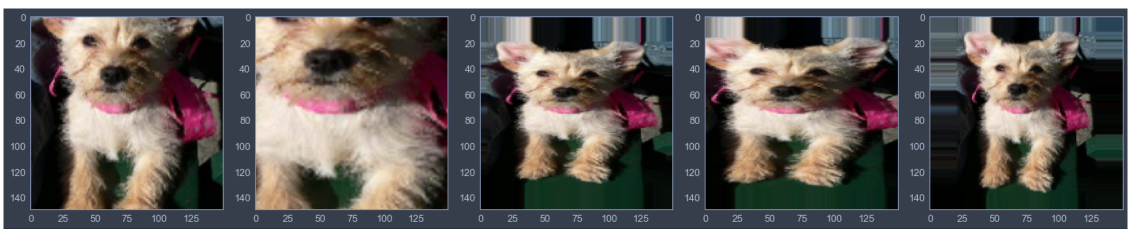
Zoom

```
[16]: image_gen = ImageDataGenerator(rescale=1./255, zoom_range=0.5)

      train_data_gen = image_gen.flow_from_directory(batch_size=BATCH_SIZE,
                                                      directory=train_dir,
                                                      shuffle=True,
                                                      target_size=(IMG_SHAPE,␣
       ↪IMG_SHAPE))
```

```
Found 2000 images belonging to 2 classes.
```

```
[17]: augmented_images = [train_data_gen[0][0][0] for i in range(5)]
      plotImages(augmented_images)
```



1.4.2 Combining together

```
[25]: image_gen_train = ImageDataGenerator(
          rescale=1./255,
          rotation_range=40,
          width_shift_range=0.2,
          height_shift_range=0.2,
          shear_range=0.2,
          zoom_range=0.2,
          horizontal_flip=True,
          fill_mode='nearest')
```

```
train_data_gen = image_gen_train.flow_from_directory(batch_size=BATCH_SIZE,
                                                     directory=train_dir,
                                                     shuffle=True,
                                                     target_size=(IMG_SHAPE,␣
 ↪IMG_SHAPE),
                                                     class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
```

1.4.3 Validation Data Generator

This will not have any image augmentation

[19]:
```
image_gen_val = ImageDataGenerator(rescale=1./255)

val_data_gen = image_gen_val.flow_from_directory(batch_size=BATCH_SIZE,
                                                 directory=validation_dir,
                                                 target_size=(IMG_SHAPE,␣
 ↪IMG_SHAPE),
                                                 class_mode='binary')
```

```
Found 1000 images belonging to 2 classes.
```

1.5 Building the model

[28]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150,␣
 ↪3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])
```

[29]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               3211776
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
_________________________________________________________________
```

[30]:
```
epochs=30
history = model.fit(
            train_data_gen,
            steps_per_epoch=int(np.ceil(total_train / float(BATCH_SIZE))),
            epochs=epochs,
            validation_data=val_data_gen,
            validation_steps=int(np.ceil(total_val / float(BATCH_SIZE)))
)
```

```
Epoch 1/30
20/20 [==============================] - 42s 2s/step - loss: 0.7402 - accuracy:
0.4965 - val_loss: 0.6909 - val_accuracy: 0.5100
Epoch 2/30
```

```
20/20 [==============================] - 32s 2s/step - loss: 0.6897 - accuracy:
0.5380 - val_loss: 0.6881 - val_accuracy: 0.5470
Epoch 3/30
20/20 [==============================] - 32s 2s/step - loss: 0.6864 - accuracy:
0.5350 - val_loss: 0.6624 - val_accuracy: 0.6030
Epoch 4/30
20/20 [==============================] - 32s 2s/step - loss: 0.6761 - accuracy:
0.5770 - val_loss: 0.6442 - val_accuracy: 0.6520
Epoch 5/30
20/20 [==============================] - 32s 2s/step - loss: 0.6615 - accuracy:
0.5990 - val_loss: 0.6396 - val_accuracy: 0.6290
Epoch 6/30
20/20 [==============================] - 32s 2s/step - loss: 0.6682 - accuracy:
0.5965 - val_loss: 0.6574 - val_accuracy: 0.6120
Epoch 7/30
20/20 [==============================] - 32s 2s/step - loss: 0.6617 - accuracy:
0.6065 - val_loss: 0.6321 - val_accuracy: 0.6370
Epoch 8/30
20/20 [==============================] - 32s 2s/step - loss: 0.6579 - accuracy:
0.6130 - val_loss: 0.6293 - val_accuracy: 0.6560
Epoch 9/30
20/20 [==============================] - 32s 2s/step - loss: 0.6438 - accuracy:
0.6190 - val_loss: 0.6312 - val_accuracy: 0.6380
Epoch 10/30
20/20 [==============================] - 32s 2s/step - loss: 0.6404 - accuracy:
0.6200 - val_loss: 0.6623 - val_accuracy: 0.5930
Epoch 11/30
20/20 [==============================] - 31s 2s/step - loss: 0.6375 - accuracy:
0.6240 - val_loss: 0.5935 - val_accuracy: 0.6840
Epoch 12/30
20/20 [==============================] - 31s 2s/step - loss: 0.6211 - accuracy:
0.6500 - val_loss: 0.5859 - val_accuracy: 0.6920
Epoch 13/30
20/20 [==============================] - 31s 2s/step - loss: 0.5924 - accuracy:
0.6940 - val_loss: 0.5881 - val_accuracy: 0.6810
Epoch 14/30
20/20 [==============================] - 31s 2s/step - loss: 0.6005 - accuracy:
0.6625 - val_loss: 0.5986 - val_accuracy: 0.6880
Epoch 15/30
20/20 [==============================] - 31s 2s/step - loss: 0.5949 - accuracy:
0.6745 - val_loss: 0.5626 - val_accuracy: 0.7110
Epoch 16/30
20/20 [==============================] - 31s 2s/step - loss: 0.5706 - accuracy:
0.6955 - val_loss: 0.5245 - val_accuracy: 0.7360
Epoch 17/30
20/20 [==============================] - 31s 2s/step - loss: 0.5621 - accuracy:
0.7135 - val_loss: 0.5484 - val_accuracy: 0.7180
Epoch 18/30
```

```
20/20 [==============================] - 31s 2s/step - loss: 0.5600 - accuracy:
0.7115 - val_loss: 0.5519 - val_accuracy: 0.7160
Epoch 19/30
20/20 [==============================] - 31s 2s/step - loss: 0.5719 - accuracy:
0.7005 - val_loss: 0.5506 - val_accuracy: 0.7350
Epoch 20/30
20/20 [==============================] - 31s 2s/step - loss: 0.5480 - accuracy:
0.7345 - val_loss: 0.5698 - val_accuracy: 0.7180
Epoch 21/30
20/20 [==============================] - 31s 2s/step - loss: 0.5519 - accuracy:
0.7075 - val_loss: 0.5137 - val_accuracy: 0.7480
Epoch 22/30
20/20 [==============================] - 31s 2s/step - loss: 0.5378 - accuracy:
0.7250 - val_loss: 0.5053 - val_accuracy: 0.7390
Epoch 23/30
20/20 [==============================] - 31s 2s/step - loss: 0.5397 - accuracy:
0.7165 - val_loss: 0.5260 - val_accuracy: 0.7460
Epoch 24/30
20/20 [==============================] - 31s 2s/step - loss: 0.5397 - accuracy:
0.7370 - val_loss: 0.5417 - val_accuracy: 0.7180
Epoch 25/30
20/20 [==============================] - 31s 2s/step - loss: 0.5290 - accuracy:
0.7300 - val_loss: 0.5103 - val_accuracy: 0.7380
Epoch 26/30
20/20 [==============================] - 31s 2s/step - loss: 0.5286 - accuracy:
0.7340 - val_loss: 0.5313 - val_accuracy: 0.7420
Epoch 27/30
20/20 [==============================] - 31s 2s/step - loss: 0.5266 - accuracy:
0.7470 - val_loss: 0.4736 - val_accuracy: 0.7860
Epoch 28/30
20/20 [==============================] - 31s 2s/step - loss: 0.5119 - accuracy:
0.7455 - val_loss: 0.4723 - val_accuracy: 0.7670
Epoch 29/30
20/20 [==============================] - 31s 2s/step - loss: 0.5254 - accuracy:
0.7385 - val_loss: 0.4817 - val_accuracy: 0.7720
Epoch 30/30
20/20 [==============================] - 31s 2s/step - loss: 0.5134 - accuracy:
0.7545 - val_loss: 0.4651 - val_accuracy: 0.7800
```

[31]:
```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)
```

```
plt.figure()
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training_Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy' )
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training_Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss' )
plt.legend(loc='lower right')
plt.title('Training and Validation Loss')
```

[31]: Text(0.5, 1.0, 'Training and Validation Loss')