In [1]:

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

import keras
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import Adam, RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
```

```
Using TensorFlow backend.
```

In [2]:

```
from keras.datasets import mnist
```

In [3]:

```
import os
PROJECT_FOLDER = os.path.join(os.path.dirname(os.getcwd()), "storage")
os.chdir(PROJECT_FOLDER)
print(os.getcwd())
```

```
D:\PyProjects\Machine-Learning-Tech\storage
```

In [4]:

```
train_path = os.path.join("dogs-vs-cats(smaller)", "train")
valid_path = os.path.join("dogs-vs-cats(smaller)", "valid")
test_path = os.path.join("dogs-vs-cats(smaller)", "test")
```

In [ ]:

```
train_batches = ImageDataGenerator().flow_from_directory(train_path,
                                                         target_size=(224, 224),
                                                         classes=["dogs", "cats"], # define which is
                                                         batch_size=64)

valid_batches = ImageDataGenerator().flow_from_directory(valid_path,
                                                         target_size=(224, 224),
                                                         classes=["dogs", "cats"],
                                                         batch_size=4)

test_batches = ImageDataGenerator().flow_from_directory(test_path,
                                                        target_size=(224, 224),
                                                        classes=["dogs", "cats"],
                                                        batch_size=10)
```

build model

In [8]:

```
vgg16_model = keras.applications.vgg16.VGG16()
```

...

In [24]:

```
print(type(vgg16_model))

model=Sequential()
for layer in vgg16_model.layers[:-1]:
    model.add(layer)

print(type(model))
```

```
<class 'keras.engine.training.Model'>
<class 'keras.engine.sequential.Sequential'>
```

In [25]:

```
print(len(model.layers))
model.layers.pop()  # this does not work anymore
print(len(model.layers))
```

```
21
21
```

In [26]:

```
for layer in model.layers:
    layer.trainable = False
model.add(Dense(2, activation='softmax'))
```

In [27]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 8,194
Non-trainable params: 134,260,544
_________________________________________________________________
```

In [28]:

```
model.compile(Adam(lr=.0001),
              loss="categorical_crossentropy",
              metrics=["accuracy"]
             )

model.fit_generator(train_batches,
                    epochs=3,
                    steps_per_epoch=8,

                    validation_data=valid_batches,
                    validation_steps=4)
```

...

In [ ]:

```
# experience: improve batch size to enable learning.
# improve number of parameters to let it actually learn some thing
```