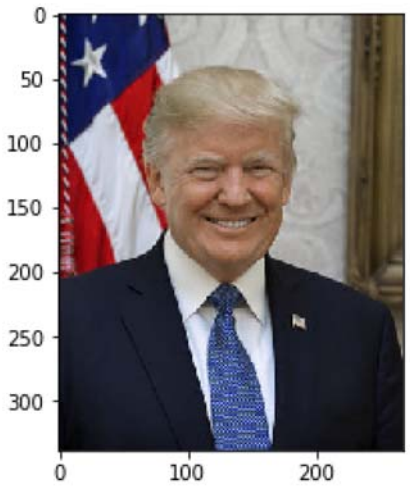
In [4]:

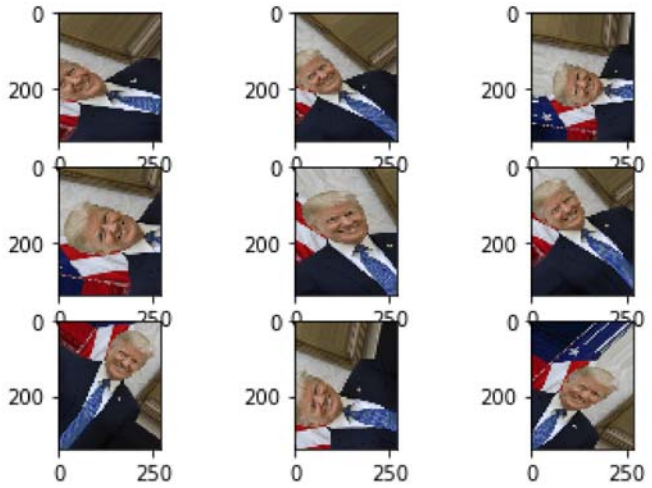
```
# %load DataAugmentation.py
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np

# 指定参数
# rotation_range 旋转
# width_shift_range 左右平移
# height_shift_range 上下平移
# zoom_range 随机放大或缩小
img_generator = ImageDataGenerator(
    rotation_range = 90,
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    zoom_range = 0.3
    )
# 导入并显示图片
img_path = 't1.jpg'
img = image.load_img(img_path)
plt.imshow(img)
plt.show()

# 将图片转为数组
x = image.img_to_array(img)
# 扩充一个维度
x = np.expand_dims(x, axis=0)
# 生成图片
gen = img_generator.flow(x, batch_size=1)

# 显示生成的图片
plt.figure()
for i in range(3):
    for j in range(3):
        x_batch = next(gen)
        idx = (3*i) + j
        plt.subplot(3, 3, idx+1)
        plt.imshow(x_batch[0]/256)
x_batch.shape
plt.show()
```





In [ ]:

In [1]:

```
# %load fasion_mnist迁移学习.py
import numpy as np
from keras.datasets import fashion_mnist
import gc

from keras.models import Sequential, Model
from keras.layers import Input, Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.applications.vgg16 import VGG16
from keras.optimizers import SGD
import matplotlib.pyplot as plt
import os

import cv2
import h5py as h5py
import numpy as np
def tran_y(y):
    y_ohe = np.zeros(10)
    y_ohe[y] = 1
    return y_ohe
epochs = 10

# 如果硬件配置较高，比如主机具备32GB以上内存，GPU具备8GB以上显存，可以适当增大这个值。VGG要求至少
ishape=48
(X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()

X_train = [cv2.cvtColor(cv2.resize(i, (ishape, ishape)), cv2.COLOR_GRAY2BGR) for i
X_train = np.concatenate([arr[np.newaxis] for arr in X_train]).astype('float32')
X_train /= 255.0

X_test = [cv2.cvtColor(cv2.resize(i, (ishape, ishape)), cv2.COLOR_GRAY2BGR) for i i
X_test = np.concatenate([arr[np.newaxis] for arr in X_test]).astype('float32')
X_test /= 255.0

y_train_ohe = np.array([tran_y(y_train[i]) for i in range(len(y_train))])
y_test_ohe = np.array([tran_y(y_test[i]) for i in range(len(y_test))])
y_train_ohe = y_train_ohe.astype('float32')
y_test_ohe = y_test_ohe.astype('float32')


model_vgg = VGG16(include_top = False, weights = 'imagenet', input_shape = (ishape,
for layer in model_vgg.layers:
        layer.trainable = False
model = Flatten()(model_vgg.output)
model = Dense(4096, activation='relu', name='fc1')(model)
model = Dense(4096, activation='relu', name='fc2')(model)
model = Dropout(0.5)(model)
model = Dense(10, activation = 'softmax', name='prediction')(model)
model_vgg_mnist_pretrain = Model(model_vgg.input, model, name = 'vgg16_pretrain')
model_vgg_mnist_pretrain.summary()
sgd = SGD(lr = 0.05, decay = 1e-5)
model_vgg_mnist_pretrain.compile(loss = 'categorical_crossentropy', optimizer = sgd
log  = model_vgg_mnist_pretrain.fit(X_train, y_train_ohe, validation_data = (X_test

score = model_vgg_mnist_pretrain.evaluate(X_test, y_test_ohe, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
/home/ian/installed/anaconda3/lib/python3.6/site-packages/h5py/__init
__.py:36: FutureWarning: Conversion of the second argument of issubdt
ype from `float` to `np.floating` is deprecated. In future, it will b
e treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 48, 48, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 48, 48, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 48, 48, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 24, 24, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 24, 24, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 24, 24, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 12, 12, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 12, 12, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 12, 12, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 6, 6, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 6, 6, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 6, 6, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 3, 3, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 3, 3, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              2101248
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
prediction (Dense)           (None, 10)                40970
=================================================================
Total params: 33,638,218
Trainable params: 18,923,530
```

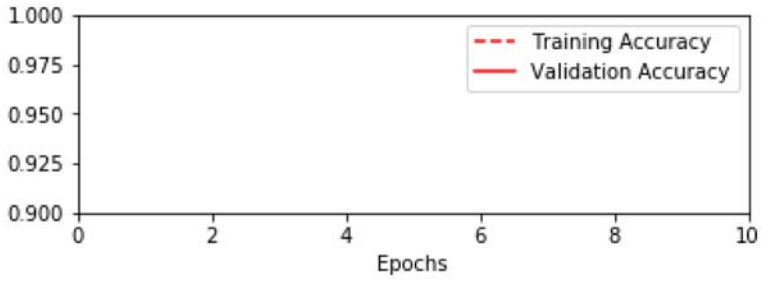
Trainable params: 10,925,950
Non-trainable params: 14,714,688

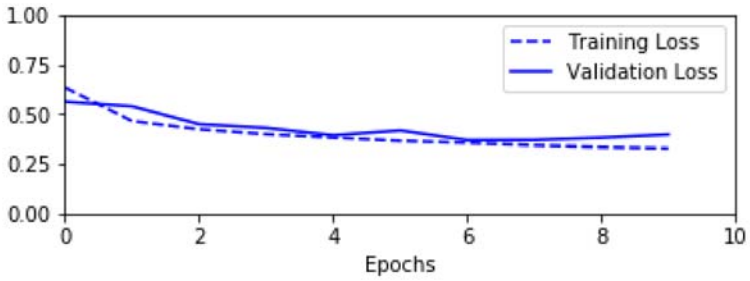
____
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 321s 5ms/step - loss:
 0.6341 - acc: 0.7722 - val_loss: 0.5626 - val_acc: 0.7988
Epoch 2/10
60000/60000 [==============================] - 323s 5ms/step - loss:
 0.4645 - acc: 0.8286 - val_loss: 0.5393 - val_acc: 0.8013
Epoch 3/10
60000/60000 [==============================] - 327s 5ms/step - loss:
 0.4231 - acc: 0.8452 - val_loss: 0.4488 - val_acc: 0.8301
Epoch 4/10
60000/60000 [==============================] - 328s 5ms/step - loss:
 0.3984 - acc: 0.8536 - val_loss: 0.4299 - val_acc: 0.8365
Epoch 5/10
60000/60000 [==============================] - 329s 5ms/step - loss:
 0.3813 - acc: 0.8587 - val_loss: 0.3927 - val_acc: 0.8567
Epoch 6/10
60000/60000 [==============================] - 331s 6ms/step - loss:
 0.3651 - acc: 0.8665 - val_loss: 0.4171 - val_acc: 0.8437
Epoch 7/10
60000/60000 [==============================] - 330s 6ms/step - loss:
 0.3552 - acc: 0.8702 - val_loss: 0.3686 - val_acc: 0.8669
Epoch 8/10
60000/60000 [==============================] - 328s 5ms/step - loss:
 0.3437 - acc: 0.8730 - val_loss: 0.3696 - val_acc: 0.8645
Epoch 9/10
60000/60000 [==============================] - 329s 5ms/step - loss:
 0.3345 - acc: 0.8773 - val_loss: 0.3810 - val_acc: 0.8598
Epoch 10/10
60000/60000 [==============================] - 334s 6ms/step - loss:
 0.3281 - acc: 0.8793 - val_loss: 0.3970 - val_acc: 0.8534
Test loss: 0.397029146361351
Test accuracy: 0.8534

In [2]:

```
plt.figure('acc')
plt.subplot(2, 1, 1)
plt.plot(log.history['acc'],'r--',label='Training Accuracy')
plt.plot(log.history['val_acc'],'r-',label='Validation Accuracy')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.axis([0, epochs, 0.9, 1])
plt.figure('loss')
plt.subplot(2, 1, 2)
plt.plot(log.history['loss'],'b--',label='Training Loss')
plt.plot(log.history['val_loss'],'b-',label='Validation Loss')
plt.legend(loc='best')
plt.xlabel('Epochs')
plt.axis([0, epochs, 0, 1])

plt.show()
```





In [ ]:

```
```