

In [1]:

```
import os
if os.path.exists('./data') == False:
    from modelarts.session import Session
    session = Session()

    session.download_data(
        bucket_path="modelarts-labs/end2end/image_recognition/dog_and_cat_25000.tar.gz",
        path="./dog_and_cat_25000.tar.gz")

    # 使用tar命令解压资源包
    !tar xf ./dog_and_cat_25000.tar.gz

    # 清理压缩包
    !rm -f ./dog_and_cat_25000.tar.gz
```

In [2]:

```
!mkdir model
```

In [3]:

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
import numpy as np

from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K
from keras.models import load_model

from keras.preprocessing.image import ImageDataGenerator
```

Using TensorFlow backend.

In [4]:

```
import os
from PIL import Image
def load_data():
    dirname = "./data"
    path = "./data"

    num_train_samples = 25000

    x_train = np.empty((num_train_samples, 224, 224, 3), dtype='uint8')
    y_train = np.empty((num_train_samples, 1), dtype='uint8')
    index = 0
    for file in os.listdir("./data"):
        image = Image.open(os.path.join(dirname, file)).resize((224, 224))
        image = np.array(image)
        x_train[index, :, :, :] = image

        if "cat" in file:
            y_train[index, 0] = 1
        elif "dog" in file:
            y_train[index, 0] = 0

        index += 1
    return (x_train, y_train)
```

In [5]:

```
(x_train, y_train) = load_data()
```

In [6]:

```
print(x_train.shape)
print(y_train.shape)
```

```
(25000, 224, 224, 3)
(25000, 1)
```

In [7]:

```
def build_model(base_model):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(2, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    print(type(model))
    return model
```

In [8]:

```
x_train,y_train= process_data(x_train,y_train)
print(x_train.shape)
print(y_train.shape)
```

NameError Traceback (most recent call last)

```
<ipython-input-8-b8227df504e8> in <module>()
----> 1 x_train,y_train= process_data(x_train,y_train)
      2 print(x_train.shape)
      3 print(y_train.shape)
```

NameError: name 'process_data' is not defined

In [9]:

```
from keras.utils import np_utils
def process_data(x_train,y_train):
    x_train = x_train.astype(np.float32)
    x_train /= 255
    n_classes = 2
    y_train = np_utils.to_categorical(y_train, n_classes)
    return x_train,y_train
```

In [10]:

```
x_train,y_train= process_data(x_train,y_train)
print(x_train.shape)
print(y_train.shape)
```

```
(25000, 224, 224, 3)
(25000, 2)
```

In [11]:

```
def build_model(base_model):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(2, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    print(type(model))
    return model
```

In [12]:

```
base_model = VGG16(weights=None, include_top=False)
```

In [13]:

```
model = build_model(base_model)
model.summary()
```

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

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0
global_average_pooling2d_1 ((None, 512)	0
dense_1 (Dense)	(None, 2)	1026
Total params: 14,715,714		
Trainable params: 14,715,714		
Non-trainable params: 0		

In [14]:

```
import keras
opt = keras.optimizers.rmsprop(lr=0.0001, decay=1e-6)
model.compile(loss='binary_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

In [15]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLRonPlateau
es = EarlyStopping(monitor='val_acc', min_delta=0.001, patience=5, verbose=1, mode='auto')
cp = ModelCheckpoint(filepath="./model/ckp_vgg16_dog_and_cat.h5", monitor="val_acc", verbose=1, save
lr = ReduceLRonPlateau(monitor="val_acc", factor=0.1, patience=3, verbose=1, mode="auto", min_lr=0)
callbacks = [es, cp, lr]
```

In [16]:

```
history = model.fit(x=x_train,
                    y=y_train,
                    batch_size=16,
                    epochs=5,
                    verbose=1,
                    callbacks=callbacks,
                    validation_split=0.25,
                    shuffle=True,
                    initial_epoch=0,
                    )
```

Train on 18750 samples, validate on 6250 samples

Epoch 1/5

18750/18750 [=====] - 174s 9ms/step - loss: 0.6764 - acc:
0.5752 - val_loss: 0.6394 - val_acc: 0.6216

Epoch 00001: val_acc improved from -inf to 0.62160, saving model to ./model/ckp_vgg16_dog_and_cat.h5

Epoch 2/5

18750/18750 [=====] - 163s 9ms/step - loss: 0.6146 - acc:
0.6647 - val_loss: 0.5746 - val_acc: 0.6714

Epoch 00002: val_acc improved from 0.62160 to 0.67136, saving model to ./model/ckp_vgg16_dog_and_cat.h5

Epoch 3/5

18750/18750 [=====] - 165s 9ms/step - loss: 0.5267 - acc:
0.7398 - val_loss: 0.7015 - val_acc: 0.7453

Epoch 00003: val_acc improved from 0.67136 to 0.74528, saving model to ./model/ckp_vgg16_dog_and_cat.h5

Epoch 4/5

18750/18750 [=====] - 166s 9ms/step - loss: 0.4242 - acc:
0.8064 - val_loss: 0.3672 - val_acc: 0.8379

Epoch 00004: val_acc improved from 0.74528 to 0.83792, saving model to ./model/ckp_vgg16_dog_and_cat.h5

Epoch 5/5

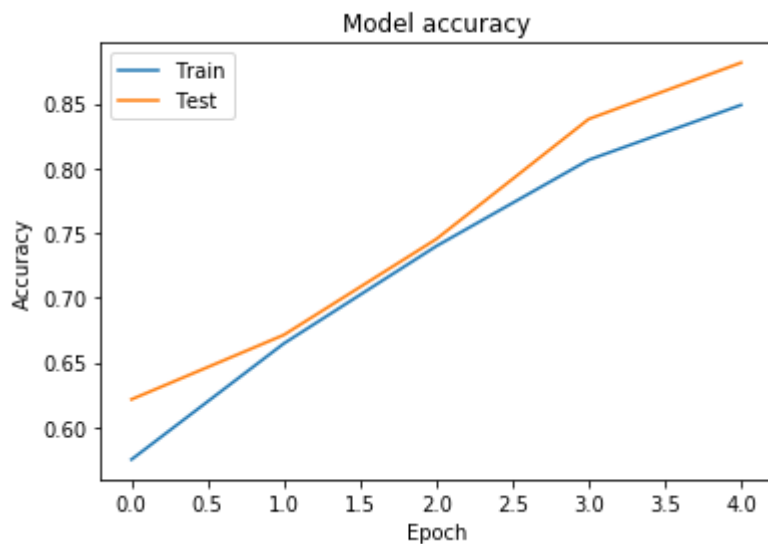
18750/18750 [=====] - 165s 9ms/step - loss: 0.3423 - acc:
0.8488 - val_loss: 0.2794 - val_acc: 0.8814

Epoch 00005: val_acc improved from 0.83792 to 0.88144, saving model to ./model/ckp_vgg16_dog_and_cat.h5

In [18]:

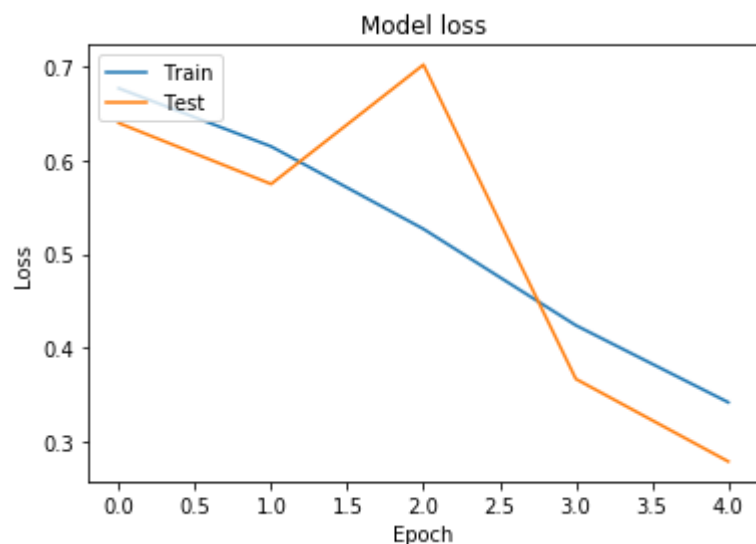
```
import matplotlib.pyplot as plt

# 绘制训练 & 验证的准确率值
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [19]:

```
# 绘制训练 & 验证的损失值
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [20]:

```
history_more_steps = model.fit(x=x_train,
                               y=y_train,
                               batch_size=16,
                               epochs=5,
                               verbose=1,
                               callbacks=callbacks,
                               validation_split=0.25,
                               shuffle=True,
                               initial_epoch=0,
                               )
```

Train on 18750 samples, validate on 6250 samples

Epoch 1/5

18750/18750 [=====] - 164s 9ms/step - loss: 0.2884 - acc:
0.8773 - val_loss: 0.2576 - val_acc: 0.8904

Epoch 00001: val_acc improved from 0.88144 to 0.89040, saving model to ./model/ckp_v
gg16_dog_and_cat.h5

Epoch 2/5

18750/18750 [=====] - 164s 9ms/step - loss: 0.2467 - acc:
0.8971 - val_loss: 0.2106 - val_acc: 0.9134

Epoch 00002: val_acc improved from 0.89040 to 0.91344, saving model to ./model/ckp_v
gg16_dog_and_cat.h5

Epoch 3/5

18750/18750 [=====] - 164s 9ms/step - loss: 0.2148 - acc:
0.9128 - val_loss: 0.2058 - val_acc: 0.9146

Epoch 00003: val_acc improved from 0.91344 to 0.91456, saving model to ./model/ckp_v
gg16_dog_and_cat.h5

Epoch 4/5

18750/18750 [=====] - 164s 9ms/step - loss: 0.1897 - acc:
0.9234 - val_loss: 0.2755 - val_acc: 0.8790

Epoch 00004: val_acc did not improve from 0.91456

Epoch 5/5

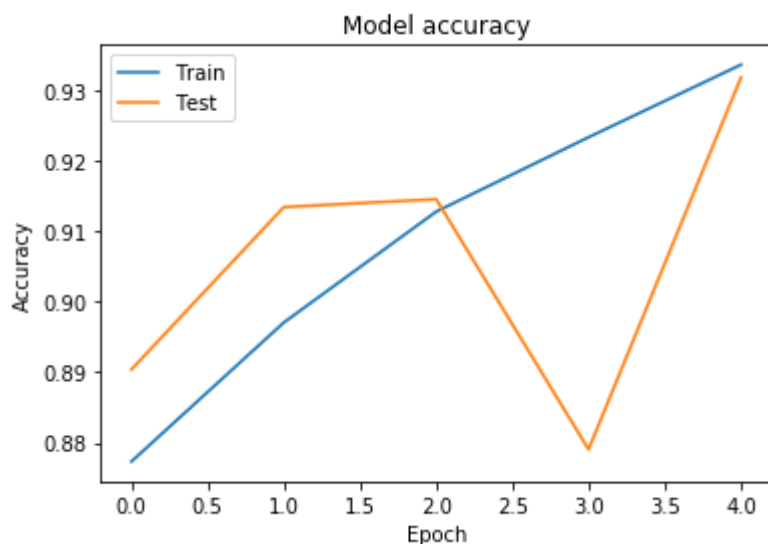
18750/18750 [=====] - 166s 9ms/step - loss: 0.1683 - acc:
0.9337 - val_loss: 0.1524 - val_acc: 0.9318

Epoch 00005: val_acc improved from 0.91456 to 0.93184, saving model to ./model/ckp_v
gg16_dog_and_cat.h5

In [21]:

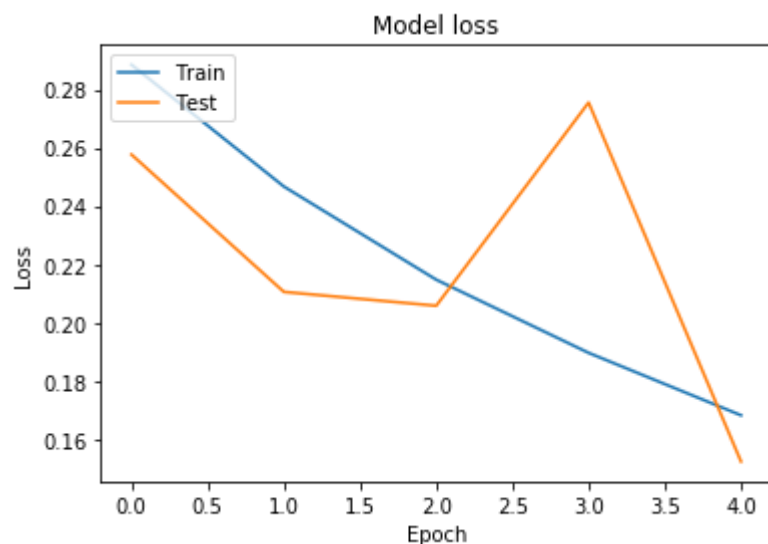
```
import matplotlib.pyplot as plt

# 绘制训练 & 验证的准确率值
plt.plot(history_more_steps.history['acc'])
plt.plot(history_more_steps.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [22]:

```
# 绘制训练 & 验证的损失值
plt.plot(history_more_steps.history['loss'])
plt.plot(history_more_steps.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [23]:

```
es = EarlyStopping(monitor='val_acc', min_delta=0.001, patience=1, verbose=1, mode='auto')
cp = ModelCheckpoint(filepath="./model/ckp_vgg16_dog_and_cat.h5", monitor="val_acc", verbose=1, save
lr = ReduceLROnPlateau(monitor="val_acc", factor=0.1, patience=3, verbose=1, mode="auto", min_lr=0)
callbacks = [es, cp, lr]
```

In [26]:

```
history_steps_15 = model.fit(x=x_train,
                             y=y_train,
                             batch_size=16,
                             epochs=20,
                             verbose=1,
                             callbacks=callbacks,
                             validation_split=0.25,
                             shuffle=True,
                             initial_epoch=0,
                             )
```

Train on 18750 samples, validate on 6250 samples

Epoch 1/20

```
18750/18750 [=====] - 162s 9ms/step - loss: 0.1393 - acc:
0.9485 - val_loss: 0.1910 - val_acc: 0.9411
```

Epoch 00001: val_acc improved from 0.93760 to 0.94112, saving model to ./model/ckp_vgg16_dog_and_cat.h5

Epoch 2/20

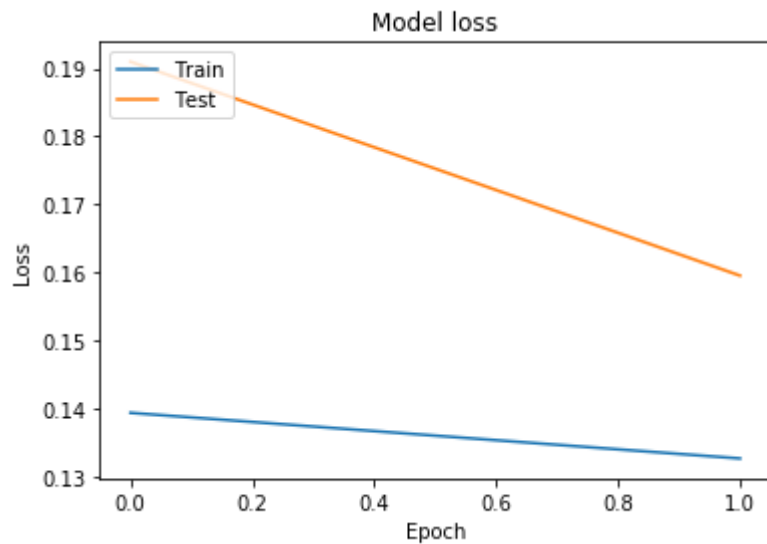
```
18750/18750 [=====] - 162s 9ms/step - loss: 0.1326 - acc:
0.9503 - val_loss: 0.1595 - val_acc: 0.9362
```

Epoch 00002: val_acc did not improve from 0.94112

Epoch 00002: early stopping

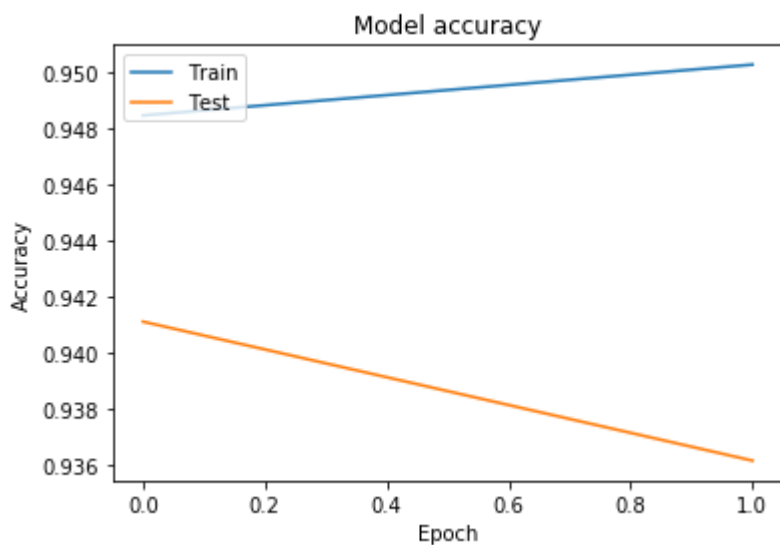
In [27]:

```
# 绘制训练 & 验证的损失值
plt.plot(history_steps_15.history['loss'])
plt.plot(history_steps_15.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [28]:

```
# 绘制训练 & 验证的准确率值
plt.plot(history_steps_15.history['acc'])
plt.plot(history_steps_15.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In []:

