In [2]:

```
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
```

```
Successfully download file modelarts-labs/end2end/image_recognition/dog_and_cat_2500
0.tar.gz from OBS to local ./dog_and_cat_25000.tar.gz
```

In [3]:

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

from keras.applications.mobilenetv2 import MobileNetV2
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
import os
from PIL import Image
```

```
Using TensorFlow backend.
```

In [4]:

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

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

In [6]:

```
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 [7]:

```
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 [8]:

```
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 [9]:

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

In [10]:

```
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 [11]:

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

In [12]:

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
es = EarlyStopping(monitor='val_acc', baseline=0.9, patience=30, verbose=1, mode='auto')
callbacks = [es]
```

In [13]:

```
history_rmsprop = model.fit(x=x_train,
                  y=y_train,
                  batch_size=32,
                  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 [==============================] - 171s 9ms/step - loss: 0.6735 - acc:
0.5806 - val_loss: 0.6449 - val_acc: 0.6312
Epoch 2/5
18750/18750 [==============================] - 157s 8ms/step - loss: 0.6285 - acc:
0.6459 - val_loss: 0.5892 - val_acc: 0.6768
Epoch 3/5
18750/18750 [==============================] - 157s 8ms/step - loss: 0.5684 - acc:
0.7079 - val_loss: 0.5238 - val_acc: 0.7426
Epoch 4/5
18750/18750 [==============================] - 156s 8ms/step - loss: 0.5147 - acc:
0.7481 - val_loss: 0.5070 - val_acc: 0.7582
Epoch 5/5
18750/18750 [==============================] - 157s 8ms/step - loss: 0.4734 - acc:
0.7770 - val_loss: 0.4430 - val_acc: 0.7939
```

In [14]:

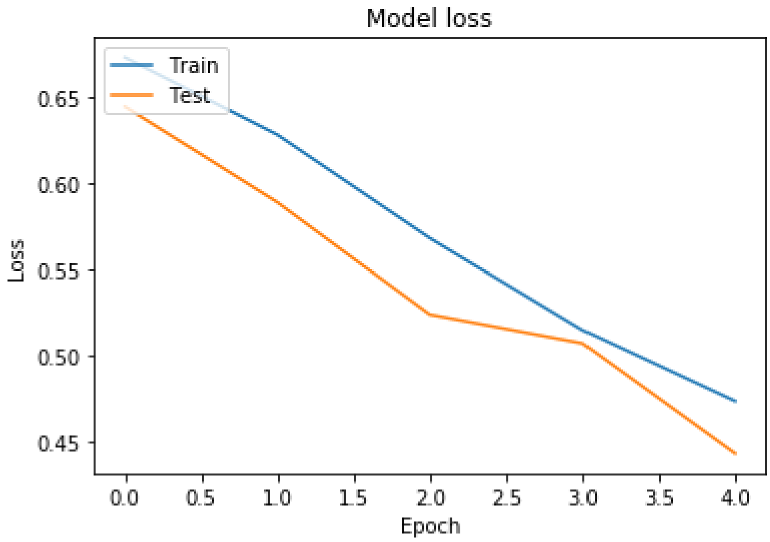
```
import matplotlib.pyplot as plt

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

```
<matplotlib.figure.Figure at 0x7f7563e6f208>
```

In [15]:

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



In [16]:

```
base_model = VGG16(weights=None, include_top=False)
model_adam = build_model(base_model)
opt = keras.optimizers.Adam(lr=0.0001, decay=1e-6)
model_adam.compile(loss=’categorical_crossentropy’,
              optimizer=opt,
              metrics=[’accuracy’])
```

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

In [17]:

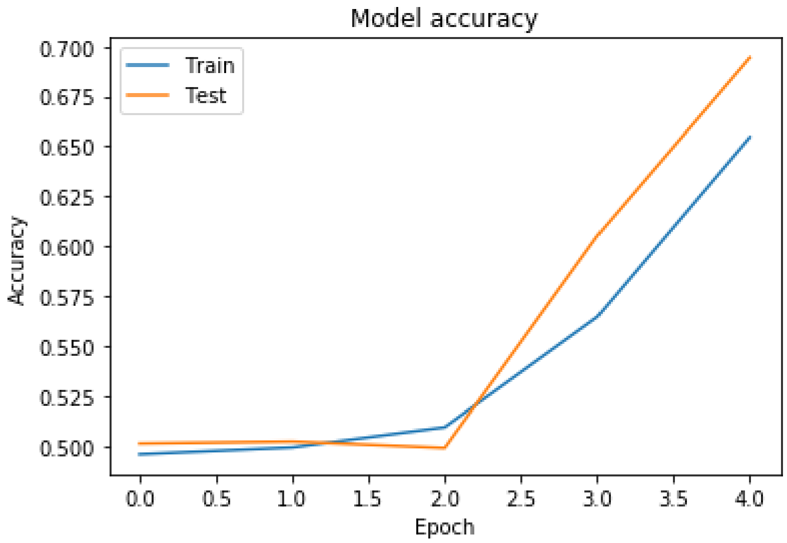
```
history_adam = model_adam.fit(x=x_train,
                    y=y_train,
                    batch_size=32,
                    epochs=5,
                    verbose=1,
                    callbacks=callbacks,
                     validation_split=0.25,
                    shuffle=True,
                    initial_epoch=0
                  )
```

```
Train on 18750 samples, validate on 6250 samples
Epoch 1/5
18750/18750 [==============================] - 155s 8ms/step - loss: 0.6933 - acc:
0.4957 - val_loss: 0.6931 - val_acc: 0.5011
Epoch 2/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6930 - acc:
0.4991 - val_loss: 0.6931 - val_acc: 0.5021
Epoch 3/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6922 - acc:
0.5092 - val_loss: 0.6931 - val_acc: 0.4989
Epoch 4/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6749 - acc:
0.5645 - val_loss: 0.6523 - val_acc: 0.6051
Epoch 5/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6139 - acc:
0.6545 - val_loss: 0.5749 - val_acc: 0.6944
```

In [18]:

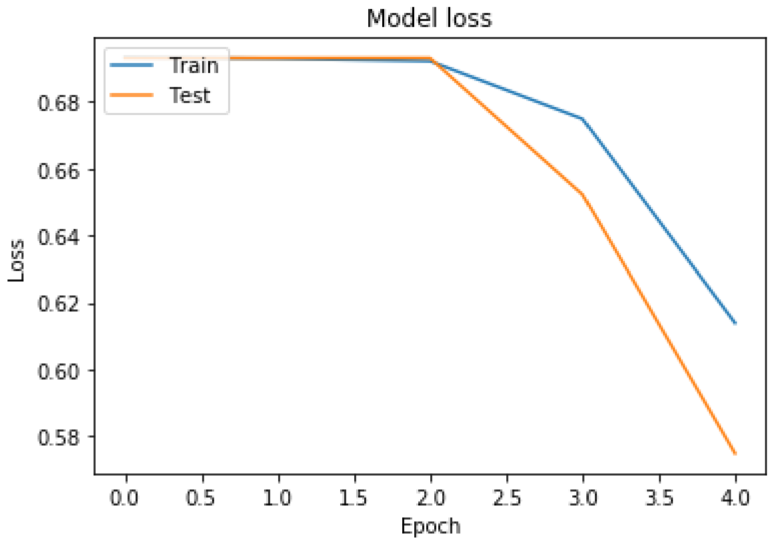
```
import matplotlib.pyplot as plt

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



In [20]:

```
base_model = VGG16(weights=None, include_top=False)
model_sgd = build_model(base_model)
opt = keras.optimizers.SGD(lr=0.0001, decay=1e-6)
model_sgd.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

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

In [21]:

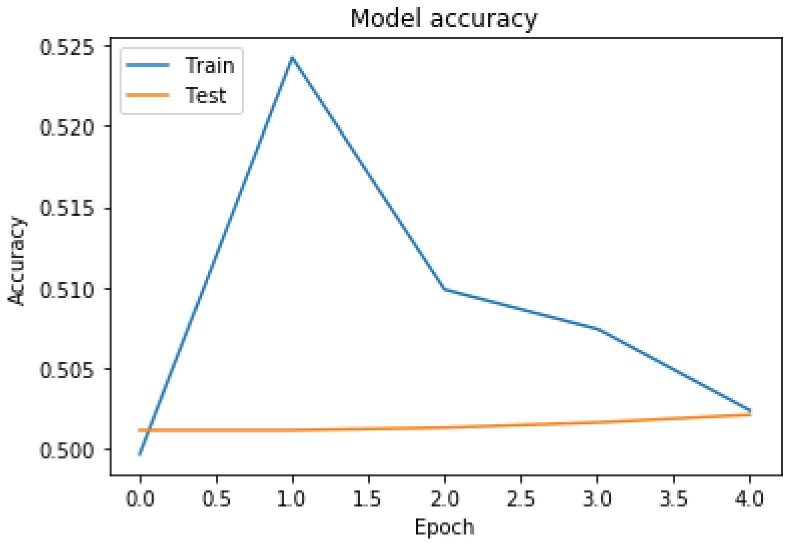
```
history_sgd = model_sgd.fit(x=x_train,
                  y=y_train,
                  batch_size=32,
                  epochs=5,
                  verbose=1,
                  callbacks=callbacks,
                  validation_split=0.25,
                  shuffle=True,
                  initial_epoch=0,
                 )
```

```
Train on 18750 samples, validate on 6250 samples
Epoch 1/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6931 - acc:
0.4996 - val_loss: 0.6931 - val_acc: 0.5011
Epoch 2/5
18750/18750 [==============================] - 153s 8ms/step - loss: 0.6931 - acc:
0.5243 - val_loss: 0.6931 - val_acc: 0.5011
Epoch 3/5
18750/18750 [==============================] - 154s 8ms/step - loss: 0.6931 - acc:
0.5099 - val_loss: 0.6931 - val_acc: 0.5013
Epoch 4/5
18750/18750 [==============================] - 155s 8ms/step - loss: 0.6931 - acc:
0.5074 - val_loss: 0.6931 - val_acc: 0.5016
Epoch 5/5
18750/18750 [==============================] - 155s 8ms/step - loss: 0.6931 - acc:
0.5023 - val_loss: 0.6931 - val_acc: 0.5021
```

In [22]:

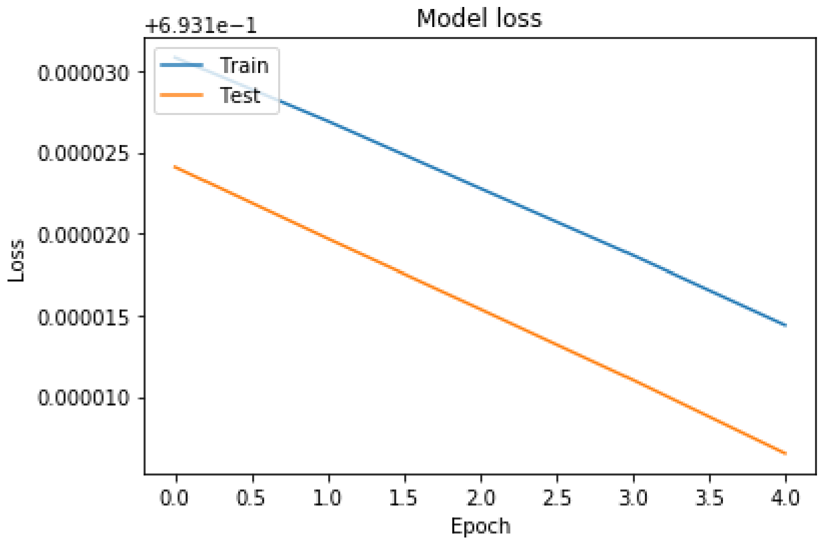
```
import matplotlib.pyplot as plt

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



In [23]:

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



In [24]:

```
base_model = VGG16(weights=None, include_top=False)
model_large_lr = build_model(base_model)
opt = keras.optimizers.Adam(lr=0.001, decay=1e-6)
model_large_lr.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

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

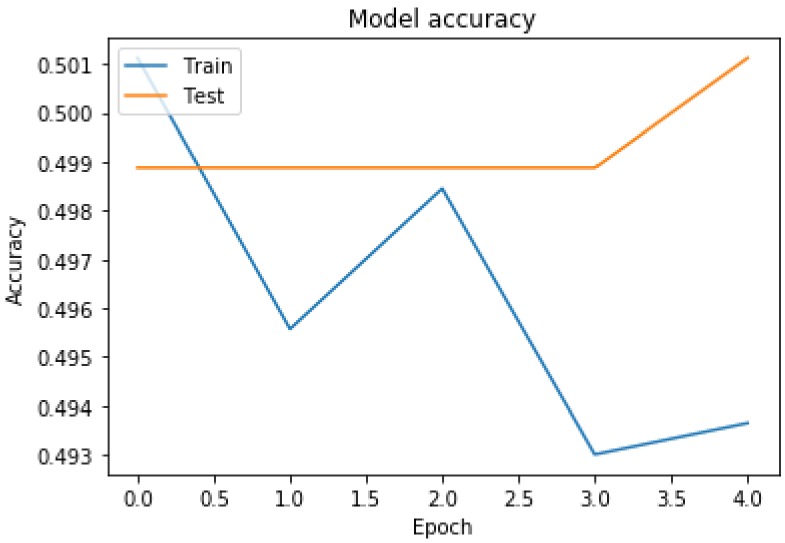
In [25]:

```
history_large_lr = model_large_lr.fit(x=x_train,
                   y=y_train,
                   batch_size=32,
                   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 [==============================] - 158s 8ms/step - loss: 0.6932 - acc:
0.5011 - val_loss: 0.6932 - val_acc: 0.4989
Epoch 2/5
18750/18750 [==============================] - 157s 8ms/step - loss: 0.6932 - acc:
0.4956 - val_loss: 0.6932 - val_acc: 0.4989
Epoch 3/5
18750/18750 [==============================] - 158s 8ms/step - loss: 0.6932 - acc:
0.4985 - val_loss: 0.6932 - val_acc: 0.4989
Epoch 4/5
18750/18750 [==============================] - 157s 8ms/step - loss: 0.6932 - acc:
0.4930 - val_loss: 0.6932 - val_acc: 0.4989
Epoch 5/5
18750/18750 [==============================] - 156s 8ms/step - loss: 0.6932 - acc:
0.4937 - val_loss: 0.6931 - val_acc: 0.5011
```

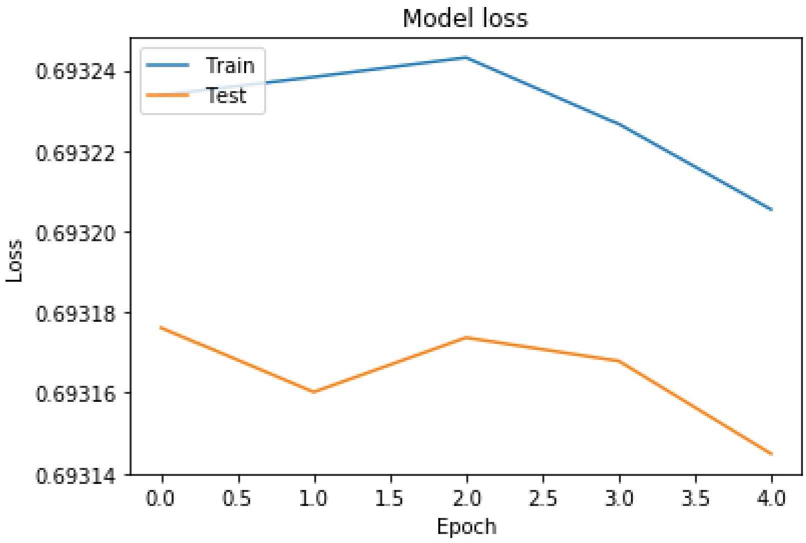
In [26]:

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



In [27]:

```
plt.plot(history_large_lr.history['loss'])
plt.plot(history_large_lr.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



In [ ]: