In [1]:
```
!pip install keras
```

```
Requirement already satisfied: keras in c:\users\shadm\anaconda3\lib\site-packages
(2.8.0)
```

In [2]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\shadm\anaconda3\lib\site-pac
kages (2.8.0)
Requirement already satisfied: keras<2.9,>=2.8.0rc0 in c:\users\shadm\anaconda3\li
b\site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\shadm\anaconda3\lib\s
ite-packages (from tensorflow) (1.6.3)
Requirement already satisfied: numpy>=1.20 in c:\users\shadm\anaconda3\lib\site-pa
ckages (from tensorflow) (1.21.5)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\shadm\anaconda3\lib
\site-packages (from tensorflow) (1.42.0)
Requirement already satisfied: protobuf>=3.9.2 in c:\users\shadm\anaconda3\lib\sit
e-packages (from tensorflow) (3.19.1)
Requirement already satisfied: flatbuffers>=1.12 in c:\users\shadm\anaconda3\lib\s
ite-packages (from tensorflow) (2.0)
Requirement already satisfied: gast>=0.2.1 in c:\users\shadm\anaconda3\lib\site-pa
ckages (from tensorflow) (0.5.3)
Requirement already satisfied: absl-py>=0.4.0 in c:\users\shadm\anaconda3\lib\site
-packages (from tensorflow) (1.0.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\sh
adm\anaconda3\lib\site-packages (from tensorflow) (0.25.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\users\shadm\anacon
da3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\shadm\anaconda3\lib\si
te-packages (from tensorflow) (1.1.0)
Requirement already satisfied: tf-estimator-nightly==2.8.0.dev2021122109 in c:\use
rs\shadm\anaconda3\lib\site-packages (from tensorflow) (2.8.0.dev2021122109)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\shadm\anaconda
3\lib\site-packages (from tensorflow) (4.1.1)
Requirement already satisfied: libclang>=9.0.1 in c:\users\shadm\anaconda3\lib\sit
e-packages (from tensorflow) (14.0.1)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\shadm\anaconda3\lib\site-
packages (from tensorflow) (1.12.1)
Requirement already satisfied: six>=1.12.0 in c:\users\shadm\anaconda3\lib\site-pa
ckages (from tensorflow) (1.16.0)
Requirement already satisfied: h5py>=2.9.0 in c:\users\shadm\anaconda3\lib\site-pa
ckages (from tensorflow) (3.6.0)
Requirement already satisfied: tensorboard<2.9,>=2.8 in c:\users\shadm\anaconda3\l
ib\site-packages (from tensorflow) (2.8.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\shadm\anaconda3\lib
\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: setuptools in c:\users\shadm\anaconda3\lib\site-pac
kages (from tensorflow) (61.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\shadm\anaconda3\lib\s
ite-packages (from tensorflow) (3.3.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\shadm\anaconda3\lib
\site-packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\s
hadm\anaconda3\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\shadm\anaconda3\lib
\site-packages (from tensorboard<2.9,>=2.8->tensorflow) (2.27.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\shadm\ana
conda3\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow) (1.8.1)
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\shadm\anaconda3\lib\s
ite-packages (from tensorboard<2.9,>=2.8->tensorflow) (2.0.3)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\shadm
\anaconda3\lib\site-packages (from tensorboard<2.9,>=2.8->tensorflow) (0.4.6)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\shadm\anaconda3\l
ib\site-packages (from tensorboard<2.9,>=2.8->tensorflow) (1.33.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\shadm\anaconda3\lib\sit
e-packages (from tensorboard<2.9,>=2.8->tensorflow) (3.3.4)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\users\shadm\anaconda3
\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow)
(4.2.2)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\shadm\anaconda3\lib\site-
```

```
packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow) (4.7.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\shadm\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.9,>=2.8->tensorflow)
(0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\shadm\anaconda
3\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.9,>=2.8-
>tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\shadm\anaconda3\li
b\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.
9,>=2.8->tensorflow) (0.4.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shadm\anaconda3\lib\site-p
ackages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow) (3.3)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\shadm\anaconda3\l
ib\site-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow) (1.
26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shadm\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow) (202
1.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\shadm\anacond
a3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.9,>=2.8->tensorflow)
(2.0.4)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\shadm\anaconda3\lib\sit
e-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tenso
rboard<2.9,>=2.8->tensorflow) (3.2.0)
```

In [3]:

```
import os

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
import seaborn as sns
from PIL import Image
np.random.seed(11)
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridS
from sklearn.metrics import accuracy_score
import itertools

import keras
from keras.utils.np_utils import to_categorical
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
from tensorflow.keras.layers import BatchNormalization
from keras.utils.np_utils import to_categorical
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.applications.resnet50 import ResNet50
from keras import backend as K
```

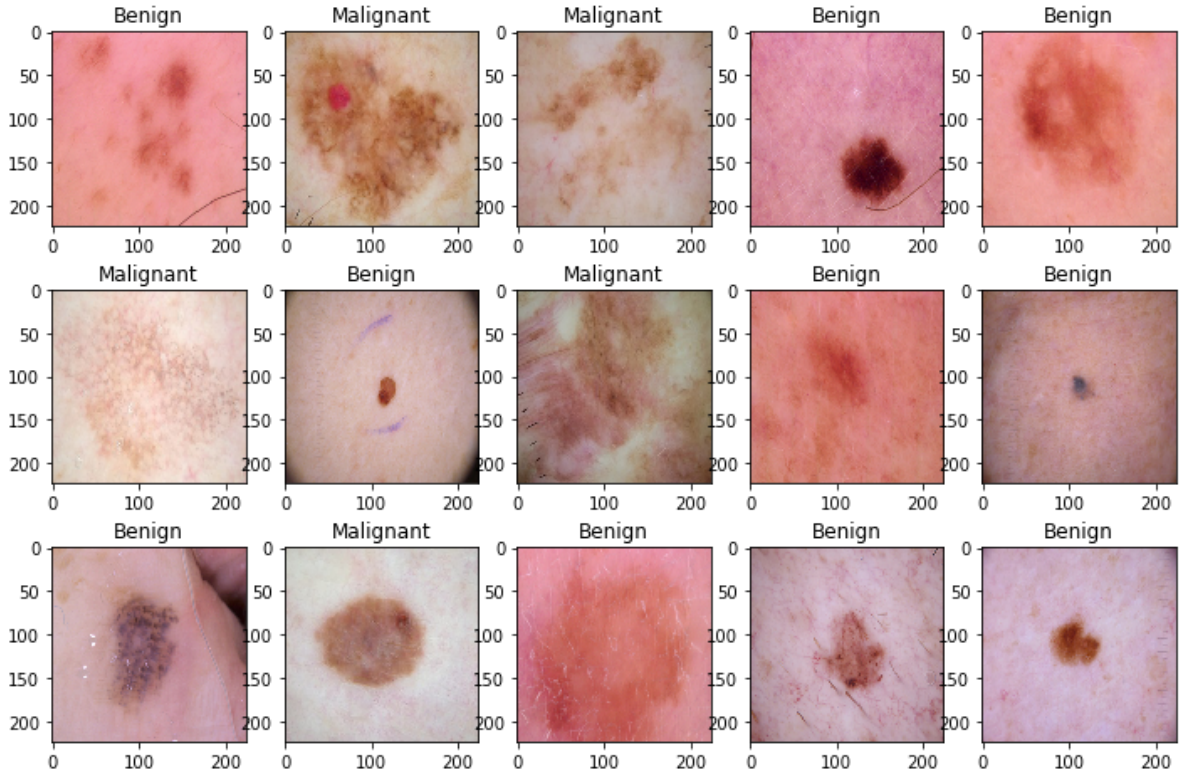
In [4]:

```
folder_benign_train = r'C:\Users\shadm\Desktop\Project\Biomedical_Image_Processing_
folder_malignant_train = r'C:\Users\shadm\Desktop\Project\Biomedical_Image_Process

folder_benign_test = r'C:\Users\shadm\Desktop\Project\Biomedical_Image_Processing_U
folder_malignant_test = r'C:\Users\shadm\Desktop\Project\Biomedical_Image_Processi

read = lambda imname: np.asarray(Image.open(imname).convert("RGB"))
```

```
ims_benign = [read(os.path.join(folder_benign_train, filename)) for filename in os
X_benign = np.array(ims_benign, dtype='uint8')
ims_malignant = [read(os.path.join(folder_malignant_train, filename)) for filename
X_malignant = np.array(ims_malignant, dtype='uint8')

ims_benign = [read(os.path.join(folder_benign_test, filename)) for filename in os.l
X_benign_test = np.array(ims_benign, dtype='uint8')
ims_malignant = [read(os.path.join(folder_malignant_test, filename)) for filename :
X_malignant_test = np.array(ims_malignant, dtype='uint8')

y_benign = np.zeros(X_benign.shape[0])
y_malignant = np.ones(X_malignant.shape[0])

y_benign_test = np.zeros(X_benign_test.shape[0])
y_malignant_test = np.ones(X_malignant_test.shape[0])

X_train = np.concatenate((X_benign, X_malignant), axis = 0)
y_train = np.concatenate((y_benign, y_malignant), axis = 0)

X_test = np.concatenate((X_benign_test, X_malignant_test), axis = 0)
y_test = np.concatenate((y_benign_test, y_malignant_test), axis = 0)

s = np.arange(X_train.shape[0])
np.random.shuffle(s)
X_train = X_train[s]
y_train = y_train[s]

s = np.arange(X_test.shape[0])
np.random.shuffle(s)
X_test = X_test[s]
y_test = y_test[s]
```

In [5]:

```
w=40
h=30
fig=plt.figure(figsize=(12, 8))
columns = 5
rows = 3

for i in range(1, columns*rows +1):
    ax = fig.add_subplot(rows, columns, i)
    if y_train[i] == 0:
        ax.title.set_text('Benign')
    else:
        ax.title.set_text('Malignant')
    plt.imshow(X_train[i], interpolation='nearest')
plt.show()
```




In [6]:
```
y_train = to_categorical(y_train, num_classes= 2)
y_test = to_categorical(y_test, num_classes= 2)
```

In [7]:
```
X_train = X_train/255.
X_test = X_test/255.
```

In [8]:
```
def build(input_shape= (224,224,3), lr = 1e-3, num_classes= 2,
          init= 'normal', activ= 'relu', optim= 'adam'):
    model = Sequential()
    model.add(Conv2D(64, kernel_size=(3, 3),padding = 'Same',input_shape=input_sha
                     activation= activ, kernel_initializer='glorot_uniform'))
    model.add(MaxPool2D(pool_size = (2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, kernel_size=(3, 3),padding = 'Same',
                     activation =activ, kernel_initializer = 'glorot_uniform'))
    model.add(MaxPool2D(pool_size = (2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer=init))
    model.add(Dense(num_classes, activation='softmax'))
    model.summary()

    if optim == 'rmsprop':
        optimizer = RMSprop(lr=lr)

    else:
        optimizer = Adam(lr=lr)

    model.compile(optimizer = optimizer ,loss = "binary_crossentropy", metrics=["a
    return model

learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy',
                                            patience=5,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=1e-7)
```

In [9]:
```
input_shape = (224,224,3)
lr = 1e-5
init = 'normal'
activ = 'relu'
optim = 'adam'
epochs = 5
batch_size = 64

model = build(lr=lr, init= init, activ= activ, optim=optim, input_shape= input_shap
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 128)               25690240

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 25,729,218
Trainable params: 25,729,218
Non-trainable params: 0
_________________________________________________________________
```

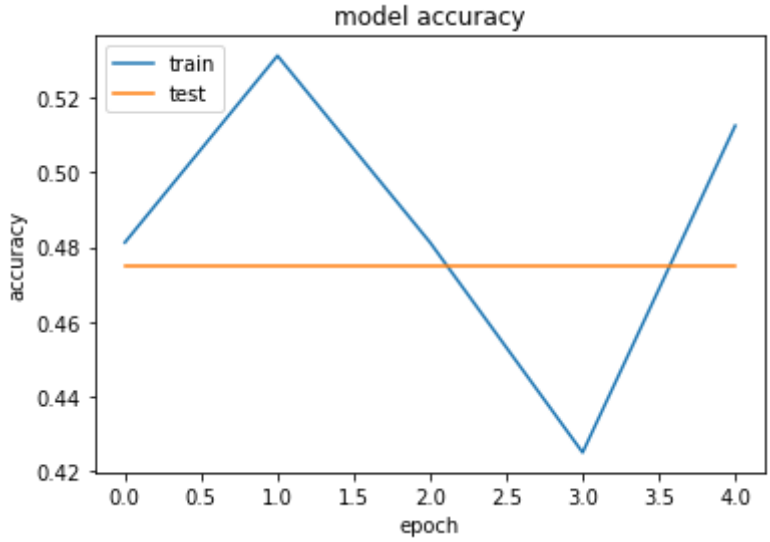
```
C:\Users\shadm\anaconda3\lib\site-packages\keras\optimizer_v2\adam.py:105: UserWar
ning: The `lr` argument is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)
```

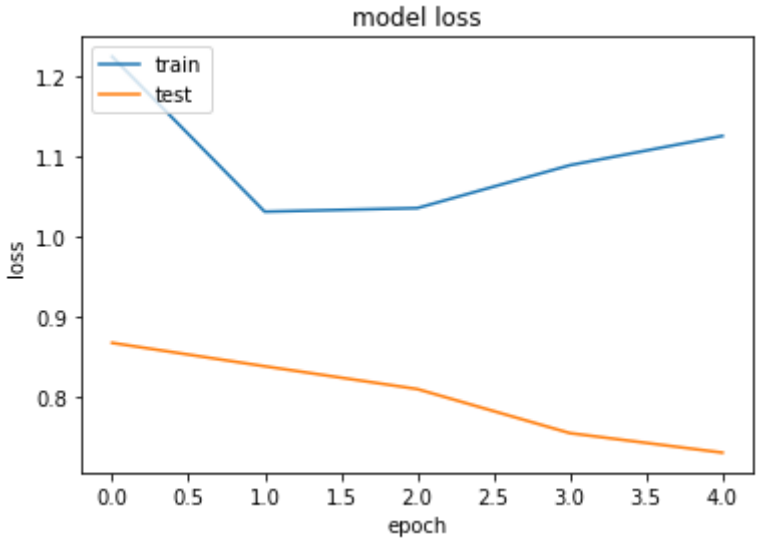
In [10]:
```
history = model.fit(X_train, y_train, validation_split=0.2,
                    epochs= epochs, batch_size= batch_size, verbose=1,
                    callbacks=[learning_rate_reduction]
                   )
```

```
Epoch 1/5
3/3 [==============================] - 14s 3s/step - loss: 1.2237 - accuracy: 0.48
12 - val_loss: 0.8665 - val_accuracy: 0.4750 - lr: 1.0000e-05
Epoch 2/5
3/3 [==============================] - 10s 3s/step - loss: 1.0302 - accuracy: 0.53
12 - val_loss: 0.8373 - val_accuracy: 0.4750 - lr: 1.0000e-05
Epoch 3/5
3/3 [==============================] - 9s 3s/step - loss: 1.0347 - accuracy: 0.481
2 - val_loss: 0.8089 - val_accuracy: 0.4750 - lr: 1.0000e-05
Epoch 4/5
3/3 [==============================] - 10s 3s/step - loss: 1.0882 - accuracy: 0.42
50 - val_loss: 0.7539 - val_accuracy: 0.4750 - lr: 1.0000e-05
Epoch 5/5
3/3 [==============================] - 9s 3s/step - loss: 1.1247 - accuracy: 0.512
5 - val_loss: 0.7295 - val_accuracy: 0.4750 - lr: 1.0000e-05
```

In [11]:
```
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
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()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```




In [12]:
```
K.clear_session()
del model
del history
```

In [13]:
```
kfold = KFold(n_splits=3, shuffle=True, random_state=11)

cvscores = []
for train, test in kfold.split(X_train, y_train):
    model = build(lr=lr,
                  init= init,
                  activ= activ,
```

```
                  optim=optim,
                  input_shape= input_shape)

    model.fit(X_train[train], y_train[train], epochs=epochs, batch_size=batch_size,
    scores = model.evaluate(X_train[test], y_train[test], verbose=0)
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
    K.clear_session()
    del model

print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 128)               25690240

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 25,729,218
Trainable params: 25,729,218
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
3/3 [==============================] - 10s 2s/step - loss: 1.3326 - accuracy: 0.57
14
Epoch 2/5
3/3 [==============================] - 8s 2s/step - loss: 1.2247 - accuracy: 0.518
8
Epoch 3/5
3/3 [==============================] - 8s 2s/step - loss: 1.2222 - accuracy: 0.488
7
Epoch 4/5
3/3 [==============================] - 9s 2s/step - loss: 1.2725 - accuracy: 0.518
8
Epoch 5/5
3/3 [==============================] - 8s 2s/step - loss: 1.0865 - accuracy: 0.511
3
accuracy: 52.24%
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 flatten (Flatten)           (None, 200704)            0
```

```
 dense (Dense)               (None, 128)               25690240

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 25,729,218
Trainable params: 25,729,218
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
3/3 [==============================] - 10s 2s/step - loss: 0.9418 - accuracy: 0.48
87
Epoch 2/5
3/3 [==============================] - 8s 2s/step - loss: 0.9604 - accuracy: 0.594
0
Epoch 3/5
3/3 [==============================] - 8s 2s/step - loss: 0.9396 - accuracy: 0.533
8
Epoch 4/5
3/3 [==============================] - 8s 2s/step - loss: 0.9063 - accuracy: 0.518
8
Epoch 5/5
3/3 [==============================] - 8s 2s/step - loss: 0.9199 - accuracy: 0.533
8
accuracy: 52.24%
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 128)               25690240

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 25,729,218
Trainable params: 25,729,218
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
3/3 [==============================] - 10s 2s/step - loss: 2.1285 - accuracy: 0.55
97
Epoch 2/5
3/3 [==============================] - 9s 2s/step - loss: 1.3042 - accuracy: 0.514
9
Epoch 3/5
3/3 [==============================] - 8s 2s/step - loss: 1.4246 - accuracy: 0.582
1
```

```
Epoch 4/5
3/3 [==============================] - 9s 2s/step - loss: 1.5070 - accuracy: 0.477
6
Epoch 5/5
3/3 [==============================] - 8s 2s/step - loss: 1.1514 - accuracy: 0.619
4
accuracy: 48.48%
50.99% (+/- 1.77%)
```

In [14]:

```
model = build(lr=lr,
              init= init,
              activ= activ,
              optim=optim,
              input_shape= input_shape)

model.fit(X_train, y_train,
          epochs=epochs, batch_size= batch_size, verbose=1,
          callbacks=[learning_rate_reduction]
         )

y_predict = model.predict_classes(X_test)

print(accuracy_score(np.argmax(y_test, axis=1),y_predict))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 dropout (Dropout)           (None, 112, 112, 64)      0

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 56, 56, 64)        0

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 128)               25690240

 dense_1 (Dense)             (None, 2)                 258

=================================================================
Total params: 25,729,218
Trainable params: 25,729,218
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
4/4 [==============================] - ETA: 0s - loss: 1.1890 - accuracy: 0.4650WA
RNING:tensorflow:Learning rate reduction is conditioned on metric `val_accuracy` w
hich is not available. Available metrics are: loss,accuracy,lr
4/4 [==============================] - 13s 3s/step - loss: 1.1890 - accuracy: 0.46
50 - lr: 1.0000e-05
Epoch 2/5
4/4 [==============================] - ETA: 0s - loss: 1.1305 - accuracy: 0.4850WA
RNING:tensorflow:Learning rate reduction is conditioned on metric `val_accuracy` w
hich is not available. Available metrics are: loss,accuracy,lr
4/4 [==============================] - 12s 3s/step - loss: 1.1305 - accuracy: 0.48
50 - lr: 1.0000e-05
Epoch 3/5
4/4 [==============================] - ETA: 0s - loss: 1.0075 - accuracy: 0.5100WA
RNING:tensorflow:Learning rate reduction is conditioned on metric `val_accuracy` w
hich is not available. Available metrics are: loss,accuracy,lr
4/4 [==============================] - 12s 3s/step - loss: 1.0075 - accuracy: 0.51
00 - lr: 1.0000e-05
Epoch 4/5
4/4 [==============================] - ETA: 0s - loss: 0.9106 - accuracy: 0.6250WA
RNING:tensorflow:Learning rate reduction is conditioned on metric `val_accuracy` w
hich is not available. Available metrics are: loss,accuracy,lr
4/4 [==============================] - 11s 3s/step - loss: 0.9106 - accuracy: 0.62
50 - lr: 1.0000e-05
Epoch 5/5
4/4 [==============================] - ETA: 0s - loss: 0.8984 - accuracy: 0.6400WA
RNING:tensorflow:Learning rate reduction is conditioned on metric `val_accuracy` w
hich is not available. Available metrics are: loss,accuracy,lr
4/4 [==============================] - 11s 3s/step - loss: 0.8984 - accuracy: 0.64
00 - lr: 1.0000e-05
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [14], in <cell line: 12>()
      1 model = build(lr=lr,
      2               init= init,
      3               activ= activ,
      4               optim=optim,
      5               input_shape= input_shape)
      7 model.fit(X_train, y_train,
      8           epochs=epochs, batch_size= batch_size, verbose=1,
      9           callbacks=[learning_rate_reduction]
     10          )
---> 12 y_predict = model.predict_classes(X_test)
     14 print(accuracy_score(np.argmax(y_test, axis=1),y_predict))

AttributeError: 'Sequential' object has no attribute 'predict_classes'
```

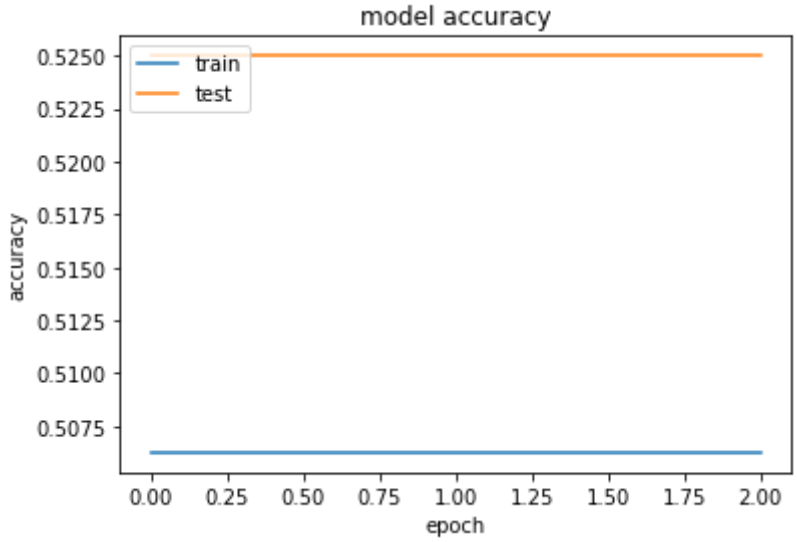
In [15]:
```
model_json = model.to_json()

with open("model.json", "w") as json_file:
    json_file.write(model_json)

model.save_weights("model.h5")
print("Saved model to disk")

del model
K.clear_session()
```

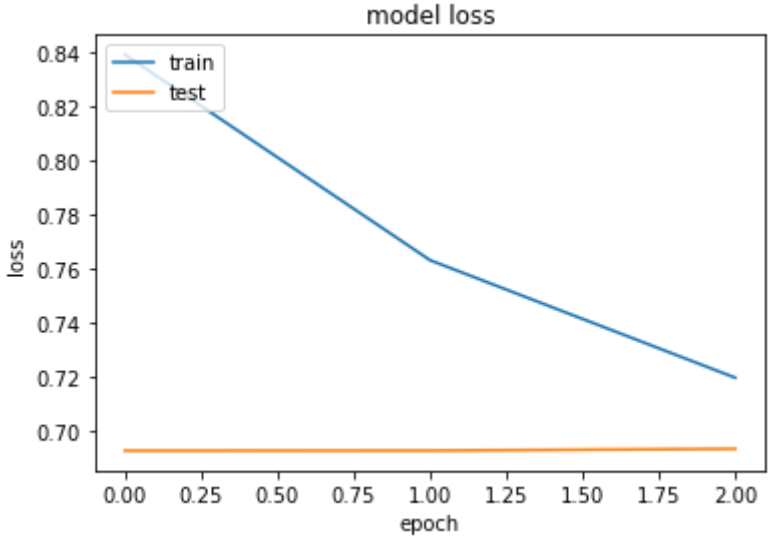
```
Saved model to disk
```

In [16]:
```
input_shape = (224,224,3)
lr = 1e-5
epochs = 3
batch_size = 64

model = ResNet50(include_top=True,
                 weights= None,
                 input_tensor=None,
                 input_shape=input_shape,
                 pooling='avg',
                 classes=2)

model.compile(optimizer = Adam(lr) ,
              loss = "binary_crossentropy",
              metrics=["accuracy"])

history = model.fit(X_train, y_train, validation_split=0.2,
                    epochs= epochs, batch_size= batch_size, verbose=1,
                    callbacks=[learning_rate_reduction]
                   )

print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
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()
```

```
Epoch 1/3
3/3 [==============================] - ETA: 0s - loss: 0.8391 - accuracy: 0.5063 W
ARNING:tensorflow:5 out of the last 15 calls to <function Model.make_test_functio
n.<locals>.test_function at 0x000001068A9A2E50> triggered tf.function retracing. T
racing is expensive and the excessive number of tracings could be due to (1) creat
ing @tf.function repeatedly in a loop, (2) passing tensors with different shapes,
(3) passing Python objects instead of tensors. For (1), please define your @tf.fun
ction outside of the loop. For (2), @tf.function has experimental_relax_shapes=Tru
e option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to https://www.tensorflow.org/guide/function#controlling_retraci
ng and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
3/3 [==============================] - 65s 16s/step - loss: 0.8391 - accuracy: 0.5
063 - val_loss: 0.6927 - val_accuracy: 0.5250 - lr: 1.0000e-05
Epoch 2/3
3/3 [==============================] - 51s 16s/step - loss: 0.7631 - accuracy: 0.5
063 - val_loss: 0.6927 - val_accuracy: 0.5250 - lr: 1.0000e-05
Epoch 3/3
3/3 [==============================] - 53s 16s/step - loss: 0.7197 - accuracy: 0.5
063 - val_loss: 0.6934 - val_accuracy: 0.5250 - lr: 1.0000e-05
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```




In [17]:

```
y_pred = model.predict(X_test)
print(accuracy_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1)))

resnet50_json = model.to_json()
```

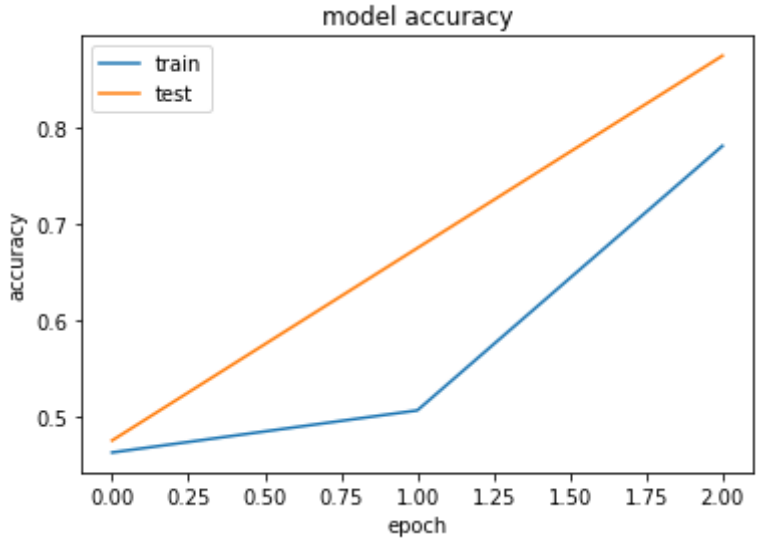
```
with open("resnet50.json", "w") as json_file:
    json_file.write(resnet50_json)

model.save_weights("resnet50.h5")
print("Saved model to disk")
```

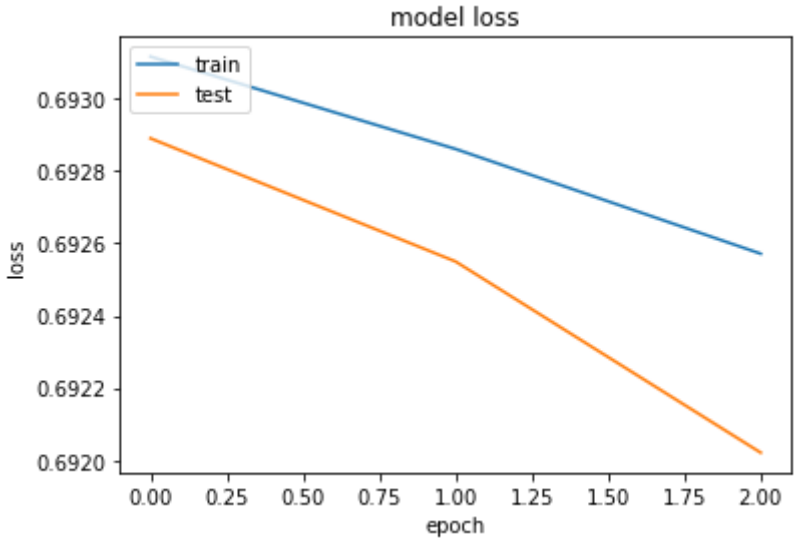
```
0.485
Saved model to disk
```

In [18]:
```
del model
K.clear_session()
```

In [19]:
```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np


input_shape = (224,224,3)
lr = 1e-5
epochs = 3
batch_size = 64

model = VGG16(include_top=True,
                 weights= None,
                 input_tensor=None,
                 input_shape=input_shape,
                 pooling='avg',
                 classes=2)

model.compile(optimizer = Adam(lr) ,
              loss = "binary_crossentropy",
              metrics=["accuracy"])

history = model.fit(X_train, y_train, validation_split=0.2,
                    epochs= epochs, batch_size= batch_size, verbose=1,
                    callbacks=[learning_rate_reduction]
                   )

print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
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()
```

```
Epoch 1/3
3/3 [==============================] - ETA: 0s - loss: 0.6931 - accuracy: 0.4625 W
ARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_functio
n.<locals>.test_function at 0x0000010672A6C550> triggered tf.function retracing. T
racing is expensive and the excessive number of tracings could be due to (1) creat
ing @tf.function repeatedly in a loop, (2) passing tensors with different shapes,
(3) passing Python objects instead of tensors. For (1), please define your @tf.fun
ction outside of the loop. For (2), @tf.function has experimental_relax_shapes=Tru
e option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to https://www.tensorflow.org/guide/function#controlling_retraci
ng and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
3/3 [==============================] - 115s 34s/step - loss: 0.6931 - accuracy: 0.
4625 - val_loss: 0.6929 - val_accuracy: 0.4750 - lr: 1.0000e-05
Epoch 2/3
3/3 [==============================] - 132s 40s/step - loss: 0.6929 - accuracy: 0.
5063 - val_loss: 0.6925 - val_accuracy: 0.6750 - lr: 1.0000e-05
Epoch 3/3
3/3 [==============================] - 111s 34s/step - loss: 0.6926 - accuracy: 0.
7812 - val_loss: 0.6920 - val_accuracy: 0.8750 - lr: 1.0000e-05
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```




In [20]:

```
y_pred = model.predict(X_test)
print(accuracy_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1)))

vgg16_json = model.to_json()

with open("vgg16.json", "w") as json_file:
    json_file.write(vgg16_json)
```

```
model.save_weights("vgg16.h5")
print("Saved model to disk")

del model
K.clear_session()
```

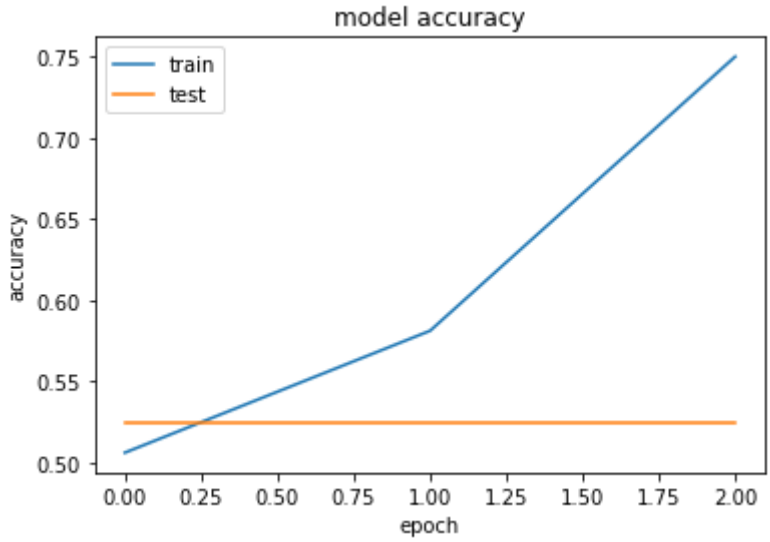
```
0.725
Saved model to disk
```

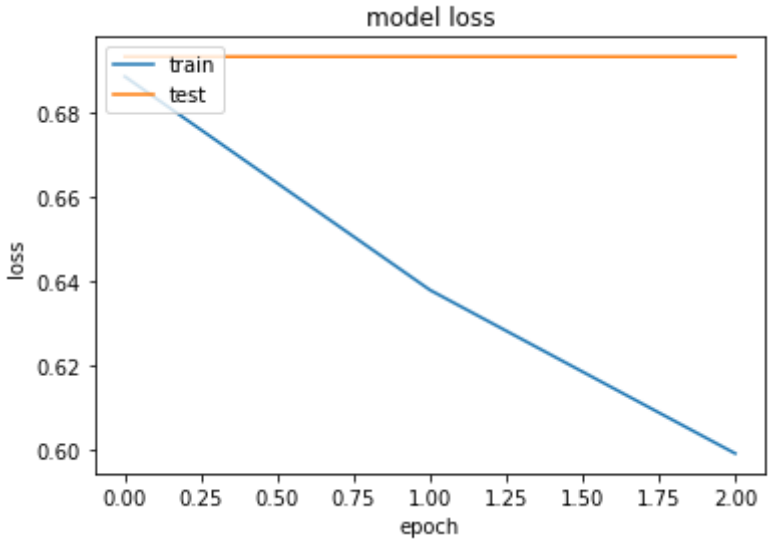
In [ ]:

In [ ]:

In [21]:
```
from keras.applications.xception import Xception


input_shape = (224,224,3)
lr = 1e-5
epochs = 3
batch_size = 64

model = Xception(include_top=True,
                 weights= None,
                 input_tensor=None,
                 input_shape=input_shape,
                 pooling='avg',
                 classes=2)

model.compile(optimizer = Adam(lr) ,
              loss = "binary_crossentropy",
              metrics=["accuracy"])

history = model.fit(X_train, y_train, validation_split=0.2,
                    epochs= epochs, batch_size= batch_size, verbose=1,
                    callbacks=[learning_rate_reduction]
                   )

print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
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()
```

```
Epoch 1/3
3/3 [==============================] - 167s 44s/step - loss: 0.6884 - accuracy: 0.
5063 - val_loss: 0.6931 - val_accuracy: 0.5250 - lr: 1.0000e-05
Epoch 2/3
3/3 [==============================] - 152s 45s/step - loss: 0.6379 - accuracy: 0.
5813 - val_loss: 0.6931 - val_accuracy: 0.5250 - lr: 1.0000e-05
Epoch 3/3
3/3 [==============================] - 172s 47s/step - loss: 0.5993 - accuracy: 0.
7500 - val_loss: 0.6931 - val_accuracy: 0.5250 - lr: 1.0000e-05
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy', 'lr'])
```




In [22]:
```
y_pred = model.predict(X_test)
print(accuracy_score(np.argmax(y_test, axis=1), np.argmax(y_pred, axis=1)))

xception_json = model.to_json()

with open("xception.json", "w") as json_file:
    json_file.write(xception_json)

model.save_weights("xception.h5")
print("Saved model to disk")

del model
K.clear_session()
```

```
0.5
Saved model to disk
```

In [ ]: