In [1]:
```
from __future__ import print_function

from sklearn.preprocessing import minmax_scale
import math
import numpy as np
from sklearn.preprocessing import LabelBinarizer
import tensorflow as tf
import numpy as np
import os
import csv
import re
import sys

import copy

#one can disable the imports below if not plotting / saving
from keras.utils import plot_model
import matplotlib.pyplot as plt

# loading data
def load_raw_data():
    ori_file = os.path.join("/home","englab5510","PycharmProjects","mmWave2","episodeData","allEpisode.np
z")
    ori_data = np.load(ori_file)

    num_train_ep = 116

    pos_mat = ori_data['position_matrix_array']
    best_ray = ori_data['best_ray_array']
    pos_mat[pos_mat < 0 ] = -1
    pos_mat[pos_mat > 0 ] = 1
    #path_gains = ori_data['path_gains_array']
    #departure_angle = ori_data['departure_angles_array']
    #arrival_angle = ori_data['arrival_angles_array']
    t1 = ori_data['t1s_array']

    #return pos_mat, best_ray, path_gains, departure_angle, arrival_angle, t1
    return pos_mat, best_ray, t1

# open validity file
def validity_check(pos_mat):
    valid_invalid_dir = os.path.join("/home","englab5510","software","MIMO_5G_Data","5gm-data-2","Valid_a
nd_Invalid_channels","list1_valids_and_invalids.csv")

    with open(valid_invalid_dir, 'r') as f:
        validity = list(csv.reader(f, delimiter=","))
    validity = np.array(validity[0:], dtype=np.str)

    # record valid and invalid cases in an array. To be used for comparison later
    array = np.zeros((pos_mat.shape[0], pos_mat.shape[1], pos_mat.shape[2]))

    for i in range(validity.shape[0]):
        episode = int(validity[i][1])
        scene = int(validity[i][2])
        receiver = int(validity[i][3])
        valid = validity[i][0]

        if re.match('V', valid, flags=0):
            array[episode][scene][receiver] = 1
        else:
            array[episode][scene][receiver] = 0

    # create a matrix to keep track of valid cases and cases where there's 50 scenes.
    validity_matrix = np.zeros((pos_mat.shape[0], pos_mat.shape[2], pos_mat.shape[1]))
    valid_count = np.zeros((pos_mat.shape[0], pos_mat.shape[2])) # to count valid scenes for each receive
r

    # loop through array variable, count valid receivers
    for index, x in np.ndenumerate(array):
        if x == 1:  # valid cases only
            valid_count[index[0], index[2]] += 1
            validity_matrix[index[0], index[2], index[1]] = 1

    # count number of times 50 scene appears for each (ep, rx) pair
    # unique, counts = np.unique(valid_count, return_counts=True)
    # dictionary = dict(zip(unique, counts))
    # print(dictionary)

    return valid_count, validity_matrix

pos_mat, best_ray, t1 = load_raw_data()
valid_count, validity_matrix = validity_check(pos_mat)
```

```
Using TensorFlow backend.
```

In [2]:
```
def get_valid_data(pos_mat, best_ray, t1, valid_count, validity_matrix):

    position_matrix_all = []
    best_ray_all = []
    t1_all = []

    for i in range(0, pos_mat.shape[0]): # ep
        for k in range(0, pos_mat.shape[2]): # rx
            if valid_count[i, k] >= 1:
                temp = np.where(validity_matrix[i,k,:] != 0) # validity_matrix:(ep, rx, scene)

                best_ray_data = best_ray[i,temp,k]
                best_ray_data = best_ray_data.reshape(best_ray_data.shape[1], best_ray_data.shape[2])

                if np.any(np.argwhere(np.isnan(best_ray_data))):
                    continue

                position_data = pos_mat[i,temp,k,:,:]
                position_data = position_data.reshape(position_data.shape[1], position_data.shape[2], pos
ition_data.shape[3])
                position_data = position_data.reshape(position_data.shape[0], position_data.shape[1], pos
ition_data.shape[2], 1)
                t1_data = t1[i,temp,k,:,:]
                t1_data = t1_data.reshape(t1_data.shape[1], t1_data.shape[2], t1_data.shape[3]) # scene,
 16, 16
                t1_data = t1_data.reshape(t1_data.shape[0], -1)


                while (best_ray_data.shape[0] < 50):
                    position_data = np.insert(position_data, [0], position_data[0,:,:], axis=0)
                    best_ray_data = np.insert(best_ray_data, [0], best_ray_data[0,:], axis=0)
                    t1_data = np.insert(t1_data, [0], t1_data[0,:], axis=0)

                position_matrix_all.append(position_data)
                best_ray_all.append(best_ray_data)
                t1_all.append(t1_data)
                #print(position_data.shape, best_ray_data.shape, t1_data.shape)


    # convert all to np array, reshape into (num_valid_cases of (ep,rx), scene_num, 23000) for pos matrix
    position_matrix_all = np.array(position_matrix_all)
    best_ray_all = np.array(best_ray_all)
    t1_all = np.array(t1_all)

    # reshape all data
    numUPAAntennaElements = 4*4

    #convert output (i,j) to single number (the class label) and eliminate pairs that do not appear
    temp = best_ray_all.reshape((-1,2))
    full_y = (best_ray_all[:,:,0] * numUPAAntennaElements + best_ray_all[:,:,1]).astype(np.int)
    temp = (temp[:,0] * numUPAAntennaElements + temp[:,1]).astype(np.int)

    classes = set(temp)
    y_train = np.zeros([best_ray_all.shape[0], best_ray_all.shape[1]])

    t1_data_valid = np.zeros((best_ray_all.shape[0], best_ray_all.shape[1], len(classes)))

    for idx, cl in enumerate(classes): #map in single index, cl is the original class number, idx is its
 index
        t1_data_valid[:,:,idx] = t1_all[:,:,cl] # extract power of valid
        cl_idx = np.nonzero(full_y == cl)
        y_train[cl_idx[0], cl_idx[1]] = idx
    ratio = [40, 45, 50]

    y_dat = np.empty((y_train.shape[0], 50, len(classes)))
    for i in range(0, y_train.shape[0]):
        y_dat[i, :] = tf.keras.utils.to_categorical(y_train[i,:], len(classes))

    print(position_matrix_all.shape, y_dat.shape)

    return position_matrix_all, y_dat

position_matrix_all, y_dat = get_valid_data(pos_mat, best_ray, t1, valid_count, validity_matrix)
```

```
(863, 50, 46, 500, 1) (863, 50, 61)
```

In [3]:
```
X = position_matrix_all.reshape((-1, 46, 500, 1))
Y = y_dat.reshape((-1, 61))
ratio = X.shape[0]*np.array([0.8, 0.1, 0.1])
ratio[1] = ratio[0]+ratio[1]
ratio[2] = ratio[1]+ratio[2]
print(X.shape, ratio)
X_train = X[0:int(ratio[0])]
X_valid = X[int(ratio[0]):int(ratio[1])]
X_test = X[int(ratio[1]):int(ratio[2])]
Y_train = Y[0:int(ratio[0])]
Y_valid = Y[int(ratio[0]):int(ratio[1])]
Y_test = Y[int(ratio[1]):int(ratio[2])]
print(np.unique(X_train))
```

```
(43150, 46, 500, 1) [34520. 38835. 43150.]
[-1  0  1]
```

In [4]:
```
from tensorflow.keras.callbacks import EarlyStopping, CSVLogger
from tensorflow.keras.callbacks import ModelCheckpoint
'''Trains a simple deep NN on ITA paper drop based dataset.

Adapted by AK: Feb 7, 2018 - I took out the graphics. Uses Pedro's datasets with 6 antenna elements per U
PA, which has 26 classes.

See for explanation about convnet and filters:
https://datascience.stackexchange.com/questions/16463/what-is-are-the-default-filters-used-by-keras-convo
lution2d
and
http://cs231n.github.io/convolutional-networks/
'''

batch_size = 32
epochs = 150

numUPAAntennaElements=4*4 #4 x 4 UPA

numClasses = Y_train.shape[1] #total number of labels

train_nexamples=X_train.shape[0]
test_nexamples=X_test.shape[0]
nrows=X_train.shape[1]
ncolumns=X_train.shape[2]

print('test_nexamples = ', test_nexamples)
print('train_nexamples = ', train_nexamples)
print('input matrices size = ', nrows, ' x ', ncolumns)
print('numClasses = ', numClasses)

#here, do not convert matrix into 1-d array
#X_train = X_train.reshape(train_nexamples,nrows*ncolumns)
#X_test = X_test.reshape(test_nexamples,nrows*ncolumns)

input_shape = (nrows, ncolumns, 1) #the input matrix with the extra dimension requested by Keras

print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
print(X_valid.shape[0], 'validation samples')
print(X_test.shape[0]+X_train.shape[0]+X_valid.shape[0], 'total samples')
print("Finished reading datasets")

# declare model Convnet with two conv1D layers following by MaxPooling layer, and two dense layers
# Dropout layer consists in randomly setting a fraction rate of input units to 0 at each update during tr
aining time, which helps prevent overfitting.
# Creates a session with log_device_placement set to True.

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Conv2D(100, kernel_size=(10,10),
            activation='relu',
             input_shape=input_shape))
model.add(tf.keras.layers.Conv2D(50, (12, 12), padding="SAME", activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(6, 6)))
model.add(tf.keras.layers.Conv2D(20, (10, 10), padding="SAME", activation='relu'))
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(4, activation='relu'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(numClasses, activation='softmax'))

model.summary()

es = EarlyStopping(monitor='val_loss', mode='min',verbose=1, patience=50)
checkpoint_path = "baseline_models/cp-{epoch:04d}.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
csv_logger = CSVLogger('baseline_models/training.log', append=True, separator=',')

cp_callback = tf.keras.callbacks.ModelCheckpoint(
    checkpoint_path, monitor='val_loss',mode='min', verbose=1, save_best_only=False)


model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adadelta(),
              metrics=['accuracy'])


history = model.fit(X_train, Y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    shuffle=True,
                    validation_data=(X_valid, Y_valid),
                   callbacks=[es, cp_callback, csv_logger])

# print results
score = model.evaluate(X_test, Y_test, verbose=0)
print(model.metrics_names)
print(score)
```

```
model.save('baseline_models/baseline_deep_ann_model.h5')

model.save_weights("baseline_models/baseline_deep_ann_model_weight.h5", overwrite=True)

with open('baseline_models/baseline_deep_ann_model_architecture.json', 'w') as f:
    f.write(model.to_json())

val_acc = history.history['val_acc']
acc = history.history['acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
np.savez('baseline_models/baseline_deep_ann_val_acc.npz',validation_acc=val_acc, testing_acc=acc, validat
ion_loss=val_loss, testing_loss=loss)

# enable if want to plot images
if True:
    # from tf.keras.utils import plot_model

    # install graphviz: sudo apt-get install graphviz and then pip install related packages
    plot_model(model, to_file='baseline_models/baseline_deep_ann_model.png', show_shapes = True)
```

```
test_nexamples =  4315
train_nexamples =  34520
input matrices size =  46  x  500
numClasses =  61
34520 train samples
4315 test samples
4315 validation samples
43150 total samples
Finished reading datasets
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 37, 491, 100)      10100     
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 37, 491, 50)       720050    
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 6, 81, 50)         0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 81, 20)         100020    
_________________________________________________________________
dropout (Dropout)            (None, 6, 81, 20)         0         
_________________________________________________________________
dense (Dense)                (None, 6, 81, 4)          84        
_________________________________________________________________
flatten (Flatten)            (None, 1944)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 61)                118645    
=================================================================
Total params: 948,899
Trainable params: 948,899
Non-trainable params: 0
_________________________________________________________________
Train on 34520 samples, validate on 4315 samples
Epoch 1/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.7934 - acc: 0.7577
Epoch 00001: saving model to baseline_models/cp-0001.ckpt
34520/34520 [==============================] - 322s 9ms/step - loss: 0.7932 - acc: 0.7578 - val_loss: 1.56
07 - val_acc: 0.6267
Epoch 2/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3925 - acc: 0.8791
Epoch 00002: saving model to baseline_models/cp-0002.ckpt
34520/34520 [==============================] - 318s 9ms/step - loss: 0.3925 - acc: 0.8791 - val_loss: 1.59
54 - val_acc: 0.6241
Epoch 3/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2861 - acc: 0.9136
Epoch 00003: saving model to baseline_models/cp-0003.ckpt
34520/34520 [==============================] - 322s 9ms/step - loss: 0.2859 - acc: 0.9137 - val_loss: 1.71
33 - val_acc: 0.6192
Epoch 4/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2320 - acc: 0.9282
Epoch 00004: saving model to baseline_models/cp-0004.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.2319 - acc: 0.9282 - val_loss: 1.70
48 - val_acc: 0.6007
Epoch 5/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2025 - acc: 0.9376
Epoch 00005: saving model to baseline_models/cp-0005.ckpt
34520/34520 [==============================] - 323s 9ms/step - loss: 0.2024 - acc: 0.9377 - val_loss: 2.00
75 - val_acc: 0.6343
Epoch 6/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1831 - acc: 0.9436
Epoch 00006: saving model to baseline_models/cp-0006.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.1830 - acc: 0.9435 - val_loss: 2.01
60 - val_acc: 0.6178
Epoch 7/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1661 - acc: 0.9478
Epoch 00007: saving model to baseline_models/cp-0007.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.1660 - acc: 0.9479 - val_loss: 2.01
91 - val_acc: 0.5921
Epoch 8/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1488 - acc: 0.9525
Epoch 00008: saving model to baseline_models/cp-0008.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.1488 - acc: 0.9524 - val_loss: 2.03
10 - val_acc: 0.6239
Epoch 9/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1398 - acc: 0.9560
Epoch 00009: saving model to baseline_models/cp-0009.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.1405 - acc: 0.9559 - val_loss: 1.69
27 - val_acc: 0.6049
Epoch 10/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1277 - acc: 0.9586
Epoch 00010: saving model to baseline_models/cp-0010.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.1279 - acc: 0.9586 - val_loss: 1.89
99 - val_acc: 0.5995
Epoch 11/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1206 - acc: 0.9617
Epoch 00011: saving model to baseline_models/cp-0011.ckpt
34520/34520 [==============================] - 305s 9ms/step - loss: 0.1206 - acc: 0.9617 - val_loss: 1.94
40 - val_acc: 0.6134
```

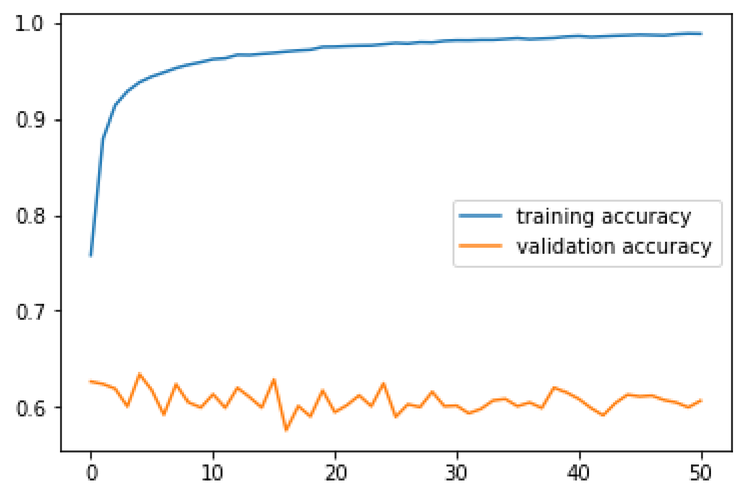
```
Epoch 12/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1159 - acc: 0.9627
Epoch 00012: saving model to baseline_models/cp-0012.ckpt
34520/34520 [==============================] - 302s 9ms/step - loss: 0.1160 - acc: 0.9627 - val_loss: 2.03
98 - val_acc: 0.5993
Epoch 13/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1098 - acc: 0.9662
Epoch 00013: saving model to baseline_models/cp-0013.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.1098 - acc: 0.9663 - val_loss: 2.64
11 - val_acc: 0.6202
Epoch 14/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1040 - acc: 0.9660
Epoch 00014: saving model to baseline_models/cp-0014.ckpt
34520/34520 [==============================] - 302s 9ms/step - loss: 0.1041 - acc: 0.9660 - val_loss: 2.30
30 - val_acc: 0.6104
Epoch 15/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.1013 - acc: 0.9673
Epoch 00015: saving model to baseline_models/cp-0015.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.1014 - acc: 0.9673 - val_loss: 2.49
69 - val_acc: 0.5995
Epoch 16/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0968 - acc: 0.9683
Epoch 00016: saving model to baseline_models/cp-0016.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0968 - acc: 0.9683 - val_loss: 2.26
57 - val_acc: 0.6290
Epoch 17/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0923 - acc: 0.9697
Epoch 00017: saving model to baseline_models/cp-0017.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0927 - acc: 0.9696 - val_loss: 2.06
91 - val_acc: 0.5759
Epoch 18/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0893 - acc: 0.9708
Epoch 00018: saving model to baseline_models/cp-0018.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0894 - acc: 0.9707 - val_loss: 2.35
57 - val_acc: 0.6012
Epoch 19/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0868 - acc: 0.9715
Epoch 00019: saving model to baseline_models/cp-0019.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0867 - acc: 0.9716 - val_loss: 2.44
90 - val_acc: 0.5903
Epoch 20/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0793 - acc: 0.9745
Epoch 00020: saving model to baseline_models/cp-0020.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0793 - acc: 0.9745 - val_loss: 2.83
58 - val_acc: 0.6174
Epoch 21/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0778 - acc: 0.9747
Epoch 00021: saving model to baseline_models/cp-0021.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0778 - acc: 0.9747 - val_loss: 2.27
72 - val_acc: 0.5947
Epoch 22/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0763 - acc: 0.9753
Epoch 00022: saving model to baseline_models/cp-0022.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0762 - acc: 0.9753 - val_loss: 2.66
86 - val_acc: 0.6021
Epoch 23/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0750 - acc: 0.9758
Epoch 00023: saving model to baseline_models/cp-0023.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0751 - acc: 0.9758 - val_loss: 2.27
24 - val_acc: 0.6123
Epoch 24/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0722 - acc: 0.9760
Epoch 00024: saving model to baseline_models/cp-0024.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0721 - acc: 0.9760 - val_loss: 2.73
99 - val_acc: 0.6009
Epoch 25/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0724 - acc: 0.9772
Epoch 00025: saving model to baseline_models/cp-0025.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0723 - acc: 0.9773 - val_loss: 2.58
41 - val_acc: 0.6248
Epoch 26/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0674 - acc: 0.9786
Epoch 00026: saving model to baseline_models/cp-0026.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0676 - acc: 0.9785 - val_loss: 2.54
11 - val_acc: 0.5898
Epoch 27/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0653 - acc: 0.9780
Epoch 00027: saving model to baseline_models/cp-0027.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0654 - acc: 0.9779 - val_loss: 2.67
76 - val_acc: 0.6030
Epoch 28/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0653 - acc: 0.9794
Epoch 00028: saving model to baseline_models/cp-0028.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0653 - acc: 0.9794 - val_loss: 2.93
13 - val_acc: 0.6000
Epoch 29/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0627 - acc: 0.9791
Epoch 00029: saving model to baseline_models/cp-0029.ckpt
```

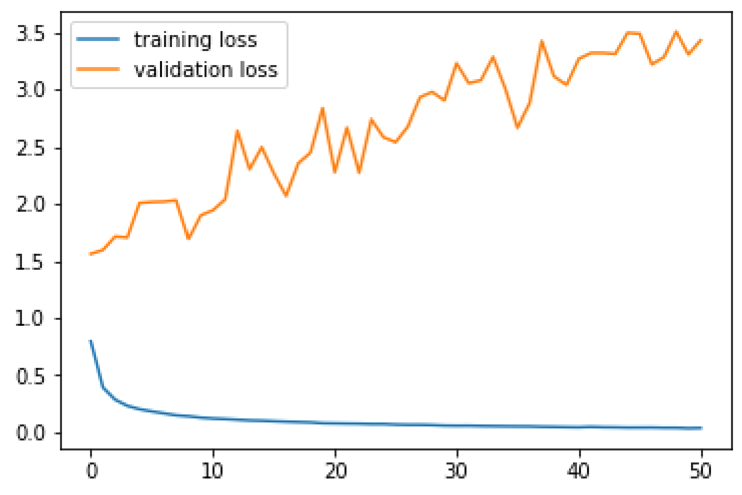
```
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0628 - acc: 0.9791 - val_loss: 2.97
80 - val_acc: 0.6160
Epoch 30/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0587 - acc: 0.9807
Epoch 00030: saving model to baseline_models/cp-0030.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0587 - acc: 0.9807 - val_loss: 2.90
30 - val_acc: 0.6009
Epoch 31/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0574 - acc: 0.9814
Epoch 00031: saving model to baseline_models/cp-0031.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0573 - acc: 0.9814 - val_loss: 3.22
90 - val_acc: 0.6016
Epoch 32/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0577 - acc: 0.9813
Epoch 00032: saving model to baseline_models/cp-0032.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0577 - acc: 0.9813 - val_loss: 3.05
42 - val_acc: 0.5935
Epoch 33/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0552 - acc: 0.9819
Epoch 00033: saving model to baseline_models/cp-0033.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0553 - acc: 0.9818 - val_loss: 3.07
99 - val_acc: 0.5981
Epoch 34/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0540 - acc: 0.9818
Epoch 00034: saving model to baseline_models/cp-0034.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0540 - acc: 0.9818 - val_loss: 3.28
52 - val_acc: 0.6070
Epoch 35/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0528 - acc: 0.9827
Epoch 00035: saving model to baseline_models/cp-0035.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0528 - acc: 0.9827 - val_loss: 3.00
79 - val_acc: 0.6086
Epoch 36/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0516 - acc: 0.9837
Epoch 00036: saving model to baseline_models/cp-0036.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0518 - acc: 0.9836 - val_loss: 2.66
69 - val_acc: 0.6009
Epoch 37/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0512 - acc: 0.9825
Epoch 00037: saving model to baseline_models/cp-0037.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0512 - acc: 0.9825 - val_loss: 2.87
87 - val_acc: 0.6049
Epoch 38/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0483 - acc: 0.9832
Epoch 00038: saving model to baseline_models/cp-0038.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0483 - acc: 0.9832 - val_loss: 3.42
54 - val_acc: 0.5991
Epoch 39/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0470 - acc: 0.9838
Epoch 00039: saving model to baseline_models/cp-0039.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0470 - acc: 0.9838 - val_loss: 3.11
53 - val_acc: 0.6204
Epoch 40/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0449 - acc: 0.9852
Epoch 00040: saving model to baseline_models/cp-0040.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0448 - acc: 0.9852 - val_loss: 3.03
99 - val_acc: 0.6153
Epoch 41/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0435 - acc: 0.9859
Epoch 00041: saving model to baseline_models/cp-0041.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0435 - acc: 0.9859 - val_loss: 3.26
87 - val_acc: 0.6086
Epoch 42/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0474 - acc: 0.9848
Epoch 00042: saving model to baseline_models/cp-0042.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0474 - acc: 0.9848 - val_loss: 3.31
96 - val_acc: 0.5991
Epoch 43/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0441 - acc: 0.9855
Epoch 00043: saving model to baseline_models/cp-0043.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0441 - acc: 0.9855 - val_loss: 3.31
93 - val_acc: 0.5914
Epoch 44/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0431 - acc: 0.9861
Epoch 00044: saving model to baseline_models/cp-0044.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0431 - acc: 0.9861 - val_loss: 3.31
19 - val_acc: 0.6042
Epoch 45/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0407 - acc: 0.9866
Epoch 00045: saving model to baseline_models/cp-0045.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0407 - acc: 0.9866 - val_loss: 3.49
58 - val_acc: 0.6130
Epoch 46/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0409 - acc: 0.9872
Epoch 00046: saving model to baseline_models/cp-0046.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0410 - acc: 0.9871 - val_loss: 3.48
88 - val_acc: 0.6111
Epoch 47/150
```

```
34496/34520 [============================>.] - ETA: 0s - loss: 0.0413 - acc: 0.9868
Epoch 00047: saving model to baseline_models/cp-0047.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0413 - acc: 0.9868 - val_loss: 3.21
93 - val_acc: 0.6121
Epoch 48/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0390 - acc: 0.9865
Epoch 00048: saving model to baseline_models/cp-0048.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0390 - acc: 0.9865 - val_loss: 3.28
45 - val_acc: 0.6074
Epoch 49/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0390 - acc: 0.9877
Epoch 00049: saving model to baseline_models/cp-0049.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0390 - acc: 0.9877 - val_loss: 3.50
77 - val_acc: 0.6049
Epoch 50/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0350 - acc: 0.9884
Epoch 00050: saving model to baseline_models/cp-0050.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0350 - acc: 0.9884 - val_loss: 3.30
66 - val_acc: 0.5998
Epoch 51/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.0369 - acc: 0.9882
Epoch 00051: saving model to baseline_models/cp-0051.ckpt
34520/34520 [==============================] - 301s 9ms/step - loss: 0.0369 - acc: 0.9882 - val_loss: 3.43
06 - val_acc: 0.6067
Epoch 00051: early stopping
['loss', 'acc']
[4.553361591318455, 0.584936268834844]

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-8cfe6833feee> in <module>
    106 
    107     pred_test = model.predict(X_test)
--> 108     for i in range(len(y_test)):
    109         if original_y_test[i] != np.argmax(pred_test[i]):
    110             myImage = X_test[i].reshape(nrows,ncolumns)

NameError: name 'y_test' is not defined
```

In [6]:
```
train_loss, train_acc = model.evaluate(X_train, Y_train, verbose=0)
test_loss, test_acc = model.evaluate(X_test, Y_test, verbose=0)
valid_loss, valid_acc = model.evaluate(X_valid, Y_valid, verbose=0)
print(model.metrics_names)
#print('Test loss rmse:', np.sqrt(score[0]))
#print('Test accuracy:', score[1])
print('Train: %.3f, Test: %.3f' % (train_acc, test_acc))

print("test_loss :" + str(test_loss) + "\n" + "test_acc :" + str(test_acc)  + "\n" + "valid_loss :" + str
(valid_loss) + "\n" + "valid_acc :" + str(valid_acc) + "\n" + "train_loss :" + str(train_loss) + "\n" +
"train_acc :" + str(train_acc))

import matplotlib.pyplot as plt

val_acc = history.history['val_acc']
acc = history.history['acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epoch_num = np.arange(0, len(val_acc), dtype=int)

plot1, = plt.plot(epoch_num, acc)
plot2, = plt.plot(epoch_num, val_acc)
plt.legend([plot1, plot2],['training accuracy', 'validation accuracy'])
plt.show()

plot1, = plt.plot(epoch_num, loss)
plot2, = plt.plot(epoch_num, val_loss)
plt.legend([plot1, plot2],['training loss', 'validation loss'])
plt.show()
```

```
['loss', 'acc']
Train: 0.996, Test: 0.585
test_loss :4.553361591318455
test_acc :0.584936268834844
valid_loss :3.4305848893469637
valid_acc :0.606720741599073
train_loss :0.012390050278228261
train_acc :0.995799536486766
```


training accuracy
validation accuracy


training loss
validation loss