In [7]:
```
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.npz")
    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_and_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 receiver

    # 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)
```

In [8]:
```
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 [9]:
```
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 [10]:
```
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.Dropout(0.5))
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(0.5))
model.add(tf.keras.layers.Dense(4, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
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/cp2-{epoch:04d}.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
csv_logger = CSVLogger('baseline_models/training2.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_model2.h5')

model.save_weights("baseline_models/baseline_deep_ann_model_weight2.h5", overwrite=True)

with open('baseline_models/baseline_deep_ann_model_architecture2.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_acc2.npz',validation_acc=val_acc, testing_acc=acc, valida
tion_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_model2.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_3 (Conv2D)            (None, 37, 491, 100)      10100
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 37, 491, 50)       720050
_________________________________________________________________
dropout_1 (Dropout)          (None, 37, 491, 50)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 81, 50)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 81, 20)         100020
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 81, 20)         0
_________________________________________________________________
dense_2 (Dense)              (None, 6, 81, 4)          84
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 81, 4)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1944)              0
_________________________________________________________________
dense_3 (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: 0.9679 - acc: 0.7085
Epoch 00001: saving model to baseline_models/cp2-0001.ckpt
34520/34520 [==============================] - 320s 9ms/step - loss: 0.9675 - acc: 0.7086 - val_loss: 1.33
02 - val_acc: 0.6503
Epoch 2/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.6447 - acc: 0.7930
Epoch 00002: saving model to baseline_models/cp2-0002.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.6447 - acc: 0.7930 - val_loss: 1.4
859 - val_acc: 0.6246
Epoch 3/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.5306 - acc: 0.8342
Epoch 00003: saving model to baseline_models/cp2-0003.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.5306 - acc: 0.8342 - val_loss: 1.9
075 - val_acc: 0.5900
Epoch 4/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.4737 - acc: 0.8552
Epoch 00004: saving model to baseline_models/cp2-0004.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.4738 - acc: 0.8551 - val_loss: 1.7
183 - val_acc: 0.5944
Epoch 5/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.4390 - acc: 0.8678
Epoch 00005: saving model to baseline_models/cp2-0005.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.4391 - acc: 0.8678 - val_loss: 1.7
708 - val_acc: 0.5300
Epoch 6/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.4154 - acc: 0.8765
Epoch 00006: saving model to baseline_models/cp2-0006.ckpt
34520/34520 [==============================] - 328s 9ms/step - loss: 0.4154 - acc: 0.8765 - val_loss: 1.51
62 - val_acc: 0.6158
Epoch 7/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3942 - acc: 0.8828
Epoch 00007: saving model to baseline_models/cp2-0007.ckpt
34520/34520 [==============================] - 321s 9ms/step - loss: 0.3942 - acc: 0.8828 - val_loss: 1.60
39 - val_acc: 0.5495
Epoch 8/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3799 - acc: 0.8882
Epoch 00008: saving model to baseline_models/cp2-0008.ckpt
34520/34520 [==============================] - 322s 9ms/step - loss: 0.3798 - acc: 0.8882 - val_loss: 1.65
59 - val_acc: 0.5710
Epoch 9/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3643 - acc: 0.8916
Epoch 00009: saving model to baseline_models/cp2-0009.ckpt
34520/34520 [==============================] - 322s 9ms/step - loss: 0.3642 - acc: 0.8916 - val_loss: 1.46
94 - val_acc: 0.6257
Epoch 10/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3487 - acc: 0.8953
Epoch 00010: saving model to baseline_models/cp2-0010.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3488 - acc: 0.8952 - val_loss: 1.62
96 - val_acc: 0.6000
Epoch 11/150
```

```
34496/34520 [============================>.] - ETA: 0s - loss: 0.3465 - acc: 0.8966
Epoch 00011: saving model to baseline_models/cp2-0011.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3465 - acc: 0.8966 - val_loss: 1.51
02 - val_acc: 0.6123
Epoch 12/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3323 - acc: 0.9004
Epoch 00012: saving model to baseline_models/cp2-0012.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3324 - acc: 0.9003 - val_loss: 1.86
38 - val_acc: 0.5205
Epoch 13/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3250 - acc: 0.9034
Epoch 00013: saving model to baseline_models/cp2-0013.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3250 - acc: 0.9034 - val_loss: 1.51
36 - val_acc: 0.6079
Epoch 14/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3217 - acc: 0.9043
Epoch 00014: saving model to baseline_models/cp2-0014.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3216 - acc: 0.9043 - val_loss: 1.53
13 - val_acc: 0.6162
Epoch 15/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3092 - acc: 0.9085
Epoch 00015: saving model to baseline_models/cp2-0015.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3090 - acc: 0.9086 - val_loss: 1.48
02 - val_acc: 0.6406
Epoch 16/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3068 - acc: 0.9106
Epoch 00016: saving model to baseline_models/cp2-0016.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3067 - acc: 0.9106 - val_loss: 1.62
10 - val_acc: 0.5972
Epoch 17/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.3013 - acc: 0.9117
Epoch 00017: saving model to baseline_models/cp2-0017.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.3012 - acc: 0.9118 - val_loss: 1.46
54 - val_acc: 0.6195
Epoch 18/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2999 - acc: 0.9100
Epoch 00018: saving model to baseline_models/cp2-0018.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.2998 - acc: 0.9100 - val_loss: 1.53
35 - val_acc: 0.6127
Epoch 19/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2927 - acc: 0.9138
Epoch 00019: saving model to baseline_models/cp2-0019.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.2926 - acc: 0.9139 - val_loss: 1.54
73 - val_acc: 0.6334
Epoch 20/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2908 - acc: 0.9140
Epoch 00020: saving model to baseline_models/cp2-0020.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2907 - acc: 0.9141 - val_loss: 1.6
480 - val_acc: 0.6239
Epoch 21/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2847 - acc: 0.9154
Epoch 00021: saving model to baseline_models/cp2-0021.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2846 - acc: 0.9154 - val_loss: 1.6
803 - val_acc: 0.6076
Epoch 22/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2871 - acc: 0.9153
Epoch 00022: saving model to baseline_models/cp2-0022.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2869 - acc: 0.9154 - val_loss: 1.6
186 - val_acc: 0.5861
Epoch 23/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2799 - acc: 0.9172
Epoch 00023: saving model to baseline_models/cp2-0023.ckpt
34520/34520 [==============================] - 323s 9ms/step - loss: 0.2798 - acc: 0.9172 - val_loss: 1.59
11 - val_acc: 0.6364
Epoch 24/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2773 - acc: 0.9187
Epoch 00024: saving model to baseline_models/cp2-0024.ckpt
34520/34520 [==============================] - 324s 9ms/step - loss: 0.2773 - acc: 0.9187 - val_loss: 1.61
71 - val_acc: 0.5956
Epoch 25/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2722 - acc: 0.9207
Epoch 00025: saving model to baseline_models/cp2-0025.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.2721 - acc: 0.9207 - val_loss: 1.64
19 - val_acc: 0.6174
Epoch 26/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2659 - acc: 0.9212
Epoch 00026: saving model to baseline_models/cp2-0026.ckpt
34520/34520 [==============================] - 322s 9ms/step - loss: 0.2660 - acc: 0.9212 - val_loss: 1.75
77 - val_acc: 0.6480
Epoch 27/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2660 - acc: 0.9226
Epoch 00027: saving model to baseline_models/cp2-0027.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2659 - acc: 0.9227 - val_loss: 1.9
023 - val_acc: 0.6431
Epoch 28/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2638 - acc: 0.9219
Epoch 00028: saving model to baseline_models/cp2-0028.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2636 - acc: 0.9219 - val_loss: 1.8
```

```
325 - val_acc: 0.6299
Epoch 29/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2664 - acc: 0.9220
Epoch 00029: saving model to baseline_models/cp2-0029.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2664 - acc: 0.9220 - val_loss: 2.1
284 - val_acc: 0.6531
Epoch 30/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2630 - acc: 0.9226
Epoch 00030: saving model to baseline_models/cp2-0030.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2630 - acc: 0.9226 - val_loss: 1.6
469 - val_acc: 0.6132
Epoch 31/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2570 - acc: 0.9250
Epoch 00031: saving model to baseline_models/cp2-0031.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2569 - acc: 0.9250 - val_loss: 2.0
146 - val_acc: 0.6431
Epoch 32/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2566 - acc: 0.9254
Epoch 00032: saving model to baseline_models/cp2-0032.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2567 - acc: 0.9253 - val_loss: 1.7
802 - val_acc: 0.6053
Epoch 33/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2522 - acc: 0.9266
Epoch 00033: saving model to baseline_models/cp2-0033.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2523 - acc: 0.9266 - val_loss: 1.8
235 - val_acc: 0.6392
Epoch 34/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2575 - acc: 0.9265
Epoch 00034: saving model to baseline_models/cp2-0034.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2578 - acc: 0.9266 - val_loss: 1.6
902 - val_acc: 0.6209
Epoch 35/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2535 - acc: 0.9257
Epoch 00035: saving model to baseline_models/cp2-0035.ckpt
34520/34520 [==============================] - 328s 10ms/step - loss: 0.2534 - acc: 0.9258 - val_loss: 1.9
157 - val_acc: 0.6561
Epoch 36/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2538 - acc: 0.9275
Epoch 00036: saving model to baseline_models/cp2-0036.ckpt
34520/34520 [==============================] - 328s 9ms/step - loss: 0.2537 - acc: 0.9275 - val_loss: 2.01
28 - val_acc: 0.6600
Epoch 37/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2478 - acc: 0.9266
Epoch 00037: saving model to baseline_models/cp2-0037.ckpt
34520/34520 [==============================] - 330s 10ms/step - loss: 0.2479 - acc: 0.9266 - val_loss: 1.9
240 - val_acc: 0.6489
Epoch 38/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2509 - acc: 0.9271
Epoch 00038: saving model to baseline_models/cp2-0038.ckpt
34520/34520 [==============================] - 330s 10ms/step - loss: 0.2508 - acc: 0.9271 - val_loss: 1.9
941 - val_acc: 0.6262
Epoch 39/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2448 - acc: 0.9300
Epoch 00039: saving model to baseline_models/cp2-0039.ckpt
34520/34520 [==============================] - 330s 10ms/step - loss: 0.2452 - acc: 0.9300 - val_loss: 1.7
917 - val_acc: 0.6352
Epoch 40/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2498 - acc: 0.9279
Epoch 00040: saving model to baseline_models/cp2-0040.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2499 - acc: 0.9279 - val_loss: 1.6
969 - val_acc: 0.6283
Epoch 41/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2432 - acc: 0.9289
Epoch 00041: saving model to baseline_models/cp2-0041.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2433 - acc: 0.9288 - val_loss: 1.7
014 - val_acc: 0.6457
Epoch 42/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2440 - acc: 0.9307
Epoch 00042: saving model to baseline_models/cp2-0042.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2441 - acc: 0.9307 - val_loss: 1.7
883 - val_acc: 0.6572
Epoch 43/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2461 - acc: 0.9267
Epoch 00043: saving model to baseline_models/cp2-0043.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2459 - acc: 0.9267 - val_loss: 2.3
588 - val_acc: 0.6542
Epoch 44/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2445 - acc: 0.9309
Epoch 00044: saving model to baseline_models/cp2-0044.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2445 - acc: 0.9309 - val_loss: 2.2
087 - val_acc: 0.6447
Epoch 45/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2429 - acc: 0.9288
Epoch 00045: saving model to baseline_models/cp2-0045.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2428 - acc: 0.9288 - val_loss: 2.2
481 - val_acc: 0.6667
Epoch 46/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2457 - acc: 0.9275
```

```
Epoch 00046: saving model to baseline_models/cp2-0046.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2456 - acc: 0.9275 - val_loss: 2.5
723 - val_acc: 0.6545
Epoch 47/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2445 - acc: 0.9309
Epoch 00047: saving model to baseline_models/cp2-0047.ckpt
34520/34520 [==============================] - 329s 10ms/step - loss: 0.2444 - acc: 0.9309 - val_loss: 2.2
194 - val_acc: 0.6512
Epoch 48/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2402 - acc: 0.9302
Epoch 00048: saving model to baseline_models/cp2-0048.ckpt
34520/34520 [==============================] - 328s 10ms/step - loss: 0.2404 - acc: 0.9302 - val_loss: 2.7
375 - val_acc: 0.6514
Epoch 49/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2406 - acc: 0.9303
Epoch 00049: saving model to baseline_models/cp2-0049.ckpt
34520/34520 [==============================] - 327s 9ms/step - loss: 0.2405 - acc: 0.9304 - val_loss: 2.11
19 - val_acc: 0.6605
Epoch 50/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2432 - acc: 0.9314
Epoch 00050: saving model to baseline_models/cp2-0050.ckpt
34520/34520 [==============================] - 328s 10ms/step - loss: 0.2430 - acc: 0.9315 - val_loss: 1.9
962 - val_acc: 0.6621
Epoch 51/150
34496/34520 [============================>.] - ETA: 0s - loss: 0.2403 - acc: 0.9305
Epoch 00051: saving model to baseline_models/cp2-0051.ckpt
34520/34520 [==============================] - 328s 10ms/step - loss: 0.2401 - acc: 0.9305 - val_loss: 2.3
782 - val_acc: 0.6691
Epoch 00051: early stopping
['loss', 'acc']
[2.741423658235512, 0.6528389339565126]
```

In [11]:
```
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: 0.958, Test: 0.653
test_loss :2.741423658235512
test_acc :0.6528389339565126
valid_loss :2.3781873699511418
valid_acc :0.6690614136732329
train_loss :0.1497212500669704
train_acc :0.957908458850613
```