# Alzheimers

July 9, 2019

[1]:
```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-notebook')
```

## 1 Loading the data

Our first task involves uploading the matrix of data from the excel spreadsheet containing patient IDs and their information. A lot of these values are not evaluated, so we must remove all the NaNs. Since we have to remove patients that don't have a CDR score, which will be what we use to label patients as having Alzheimer's Disease (AD) or not (NC), we lose a lot of patients from our original set. We go from 416 to 216, losing 200 patients.

[2]:
```
demographics = pd.read_excel('oasis_cross-sectional.xls', sheet_name=3) # load␣
 ↪data
print("{} columns and {} rows".format(demographics.shape[1], demographics.
 ↪shape[0]))
df = demographics.dropna(how='any') # remove NaN values
df.head()
```

```
11 columns and 416 rows
```

[2]:

| | ID | M/F | Hand | Age | Educ | SES | MMSE | CDR | eTIV | nWBV | ASF |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | OAS1_0001_MR1 | F | R | 74 | 2.0 | 3.0 | 29.0 | 0.0 | 1343.75 | 0.743 | 1.30604 |
| 1 | OAS1_0002_MR1 | F | R | 55 | 4.0 | 1.0 | 29.0 | 0.0 | 1146.59 | 0.810 | 1.53063 |
| 2 | OAS1_0003_MR1 | F | R | 73 | 4.0 | 3.0 | 27.0 | 0.5 | 1454.24 | 0.708 | 1.20682 |
| 8 | OAS1_0010_MR1 | M | R | 74 | 5.0 | 2.0 | 30.0 | 0.0 | 1636.08 | 0.689 | 1.07269 |
| 9 | OAS1_0011_MR1 | F | R | 52 | 3.0 | 2.0 | 30.0 | 0.0 | 1320.81 | 0.827 | 1.32873 |

## 2 Abbreviations

#### 2.0.1 SES(Socio Economic Status)

Education codes correspond to the following levels of education: 1: less than high school grad., 2: high school grad., 3: some college, 4: college grad., 5: beyond college.

#### 2.0.2 MMSE(Mini-Mental State Examination)

Exam conducted by the doctor where the Maximum Score is 30

#### 2.0.3 CDR(Clinical Dementia Rating)

0= nondemented; 0.5 – very mild dementia; 1 = mild dementia; 2 = moderate dementia

#### 2.0.4 eTIV(Estimated total intracranial volume)

#### 2.0.5 nWBV(Normalized whole brain volume)

#### 2.0.6 ASF(Atlas scaling factor)

All patients are right handed, and thusly is a variable we will ignore. The ASF is a unitless factor that will also be ignored. To get an idea of what each kind of patient looks like, we chose patient 58 and patient 308 as examples for AD and NC relatively. We will also ignore eTIV as it a normalized measurement that is unbiased by atrophy which is something we are concerned with.

[3]:
```
print("{} Rows and {} Columns".format(df.shape[0], df.shape[1]))
df_columns = list(demographics.columns)
X_columns = np.delete(df_columns, [6,7], None) # X matrix won't have MMSE or␣
 ↪CDR scores
Xdf = df.reindex(columns=X_columns)
X = Xdf.values # creating X
HealthyExampleIndex = 27
SickExampleIndex = 145
print(X_columns)
print(X[HealthyExampleIndex])
print(X[SickExampleIndex])
```

```
216 Rows and 11 Columns
['ID' 'M/F' 'Hand' 'Age' 'Educ' 'SES' 'eTIV' 'nWBV' 'ASF']
['OAS1_0058_MR1' 'F' 'R' 46 5.0 1.0 1584.7 0.817 1.10747]
['OAS1_0308_MR1' 'F' 'R' 78 3.0 3.0 1401.13 0.703 1.25256]
```

We see that patient 58 is female, right-handed (all patients are right handed), 46 years old, well educated (5 years of education), relatively well off (SES based on the Hollingshead Index is 1, and has a normalized whole brain volume of 0.817. On the other hand, patient 308 is female, right-handed, 78 years old, not as well educated (3 years of education), not as well off (an Hollingshead Index of 3), and has a normalized whole brain volume of 0.703.

### 2.1 Loading the Images

[4]:

```
# Define function to get list of pngs based on slice number
pngs_path='OASIS_MR1_pngs'
def getcoronalPNG(path):
    l = []
    coronalslice90_files = []
    coronalslice91_files = []
    coronalslice92_files = []
    coronalslice93_files = []
    coronalslice94_files = []
    coronalslice95_files = []
    coronalslice96_files = []
    coronalslice97_files = []
    coronalslice98_files = []
    coronalslice99_files = []

    for root, directories, filenames in os.walk(path):

        for filename in filenames:
            if ".90." in filename:
                coronalslice90_files.append(os.path.join(root, filename))
            if ".91." in filename:
                coronalslice91_files.append(os.path.join(root, filename))
            if ".92." in filename:
                coronalslice92_files.append(os.path.join(root, filename))
            if ".93." in filename:
                coronalslice93_files.append(os.path.join(root, filename))
            if ".94." in filename:
                coronalslice94_files.append(os.path.join(root, filename))
            if ".95." in filename:
                coronalslice95_files.append(os.path.join(root, filename))
            if ".96." in filename:
                coronalslice96_files.append(os.path.join(root, filename))
            if ".97." in filename:
                coronalslice97_files.append(os.path.join(root, filename))
            if ".98." in filename:
                coronalslice98_files.append(os.path.join(root, filename))
            if ".99." in filename:
                coronalslice99_files.append(os.path.join(root, filename))

    l = list(zip(coronalslice90_files, coronalslice91_files,␣
↪coronalslice92_files, coronalslice93_files, coronalslice94_files,␣
↪coronalslice95_files, coronalslice96_files, coronalslice97_files,␣
↪coronalslice98_files, coronalslice99_files))

    return ((np.asarray(l)))
```

```
coronal_files0 = getcoronalPNG(pngs_path)
print(coronal_files0)
```

```
[['OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0001_MR1\\OAS1_0001_MR1.99.png']
 ['OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0002_MR1\\OAS1_0002_MR1.99.png']
 ['OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0003_MR1\\OAS1_0003_MR1.99.png']
 ...
 ['OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0455_MR1\\OAS1_0455_MR1.99.png']
 ['OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0456_MR1\\OAS1_0456_MR1.99.png']
 ['OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.90.png'
  'OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.91.png'
  'OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.92.png' ...
  'OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.97.png'
  'OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.98.png'
  'OASIS_MR1_pngs\\OAS1_0457_MR1\\OAS1_0457_MR1.99.png']]
```

[5]:
```
coronal_X_files = np.take(coronal_files0, indices=df.index.values, axis=0) #␣
 ↪keeps the images with the same index as X matrix
coronal90loc, coronal91loc, coronal92loc, coronal93loc, coronal94loc,␣
 ↪coronal95loc, coronal96loc, coronal97loc, coronal98loc, coronal99loc =␣
 ↪zip(*coronal_X_files)
```

```
#print(coronal90loc, coronal91loc, coronal92loc, coronal93loc, coronal94loc,␣
↪coronal95loc, coronal96loc, coronal97loc, coronal98loc, coronal99loc)
```

### 2.2 Cleaning up the X variables

We then need to change the X variables into interpretable simpler ones that the algorithm can understand. We encode males as 1, and females as -1. I also went ahead and encoded the hands, but one should realize that all the patients are right handed.

[6]:
```
X_id, X_gender, X_handedness, X_age, X_education, X_SES, X_eTIV, X_nWBV, X_ASF␣
↪= zip(*X) # unzips big X matrix

def gender_translator(X_gender):
    X_gender_binary = []
    X_gender_encoded = []
    for x in X_gender:
        if x == 'M':
            X_gender_binary.append(1)
            X_gender_encoded.append([0,1])
        else:
            X_gender_binary.append(-1)
            X_gender_encoded.append([1,0])

    return(zip(X_gender_binary, X_gender_encoded)) # gives us binary and␣
↪one-hot encoded for sex

def hand_translator(X_handedness):
    X_hand_binary = []
    X_hand_encoded = []
    for x in X_handedness:
        if x == 'R':
            X_hand_binary.append(1)
            X_hand_encoded.append([0,1])
        else:
            X_hand_binary.append(-1)
            X_hand_encoded.append([1,0])

    return(zip(X_hand_binary, X_hand_encoded)) # same as above but for␣
↪handedness

X_gender_binary, X_gender_encoded = zip(*gender_translator(X_gender)) #␣
↪unzipping to get our function outputs
X_hand_binary, X_hand_encoded = zip(*hand_translator(X_handedness))
```

## 2.3 Image Processing

This function handles a little of the preprocessing we do to our images, namely reading in the image from the file locations, making them grayscale, and then scaling them from 0 to 255 to 0 to 1.

[7]:
```
import sklearn
from sklearn import cluster
def prepPNGimgs(array_of_image_paths):
    l = []
    for img_file in array_of_image_paths: # for each file in the list of images.
 ↪..
        img = cv2.imread("{}".format(img_file)) # read the image...
        img = np.array(img, dtype=np.float64) / 255
        w, h, d = original_shape = tuple(img.shape)
        assert d == 3
        image_array = np.reshape(img, (w * h, d))
        kmeans = sklearn.cluster.KMeans(n_clusters=2, random_state=0).
 ↪fit(image_array)
        labels = kmeans.predict(image_array)

        def recreate_image(codebook, labels, w, h):
            """Recreate the (compressed) image from the code book & labels"""
            d = codebook.shape[1]
            image = np.zeros((w, h, d))
            label_idx = 0
            for i in range(w):
                for j in range(h):
                    image[i][j] = (codebook[labels[label_idx]])
                    label_idx += 1
            return image

        clustImg = recreate_image(kmeans.cluster_centers_, labels, w, h)

        l.append(clustImg)

    return(np.asarray(l))
```

## 2.4 Cleaning up the Y variables

[8]:
```
Y_CDR_columns = [column_name for column_name in df_columns if column_name ==␣
 ↪'CDR']

Y_CDR_df = df.reindex(columns=Y_CDR_columns)
Y_CDR = Y_CDR_df.values

Y_MMSE_columns = [column_name for column_name in df_columns if column_name ==␣
 ↪'MMSE']
```

```python
Y_MMSE_df = df.reindex(columns=Y_MMSE_columns)
Y_MMSE = Y_MMSE_df.values

CDR_threshold_0 = 0 # threshold values by CDR scale
CDR_threshold_0point5 = 0.5
CDR_threshold_1 = 1

MMSE_threshold_24 = 24 # threshold values by MMSE scale
MMSE_threshold_18 = 18

def CDR_probable_AD_thresholder(Y_CDR, threshold_value):
    Y_CDR_binary = []
    Y_CDR_encoded = []
    for y in Y_CDR:
        if y > threshold_value:
            Y_CDR_binary.append(1)
            Y_CDR_encoded.append([0,1])
        else:
            Y_CDR_binary.append(-1)
            Y_CDR_encoded.append([1,0])

    return((zip(Y_CDR_binary, Y_CDR_encoded)))

def MMSE_probable_Dementia_thresholder(Y_MMSE, threshold_value):
    Y_MMSE_binary = []
    Y_MMSE_encoded = []
    for y in Y_MMSE:
        if y < threshold_value:
            Y_MMSE_binary.append(1)
            Y_MMSE_encoded.append([0,1])
        else:
            Y_MMSE_binary.append(-1)
            Y_MMSE_encoded.append([1,0])

    return(zip(Y_MMSE_binary, Y_MMSE_encoded))

Y_CDR_binary, Y_CDR_encoded = zip(*CDR_probable_AD_thresholder(Y_CDR,␣
 ↪CDR_threshold_0))
Y_MMSE_binary, Y_MMSE_encoded = zip(*MMSE_probable_Dementia_thresholder(Y_MMSE,␣
 ↪MMSE_threshold_24))
print("Patient 58: CDR: {} ; MMSE: {}".
 ↪format(Y_CDR[HealthyExampleIndex],Y_MMSE[HealthyExampleIndex]))
print("Patient 308: CDR: {} ; MMSE: {}".
 ↪format(Y_CDR[SickExampleIndex],Y_MMSE[SickExampleIndex]))
```

```
Patient 58: CDR: [0.] ; MMSE: [30.]
Patient 308: CDR: [2.] ; MMSE: [15.]
```

Continuing with the same examples as before, patient 58 has a CDR score of 0, which is healthy and essentially NC, and the max MMSE score of 30, also healthy. On the other hand, patient 308 has a CDR score of 2, which is the maximum score on the CDR scale, and has a MMSE score of 15, extremely unhealthy, and indicative of AD.

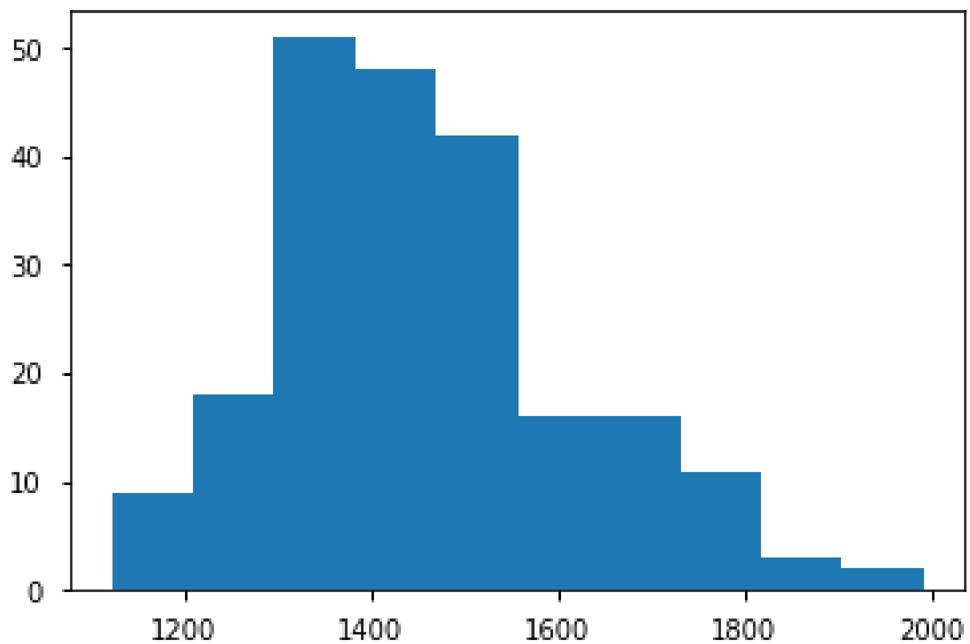
### 2.5 Final Processing Step

[9]:
```
# turning everything into numpy arrays

df_index = np.asarray(df.index.values)
X_id = np.asarray(X_id)
X_gender = np.asarray(X_gender)
X_gender_binary = np.asarray(X_gender_binary)
X_gender_encoded = np.asarray(X_gender_encoded)
X_handedness = np.asarray(X_handedness)
X_hand_binary = np.asarray(X_hand_binary)
X_hand_encoded = np.asarray(X_hand_encoded)
X_age = np.asarray(X_age)
X_education = np.asarray(X_education)
X_SES = np.asarray(X_SES)
X_eTIV = np.asarray(X_eTIV)
X_nWBV = np.asarray(X_nWBV)
X_ASF = np.asarray(X_ASF)
coronal90loc = np.asarray(coronal90loc)
coronal91loc = np.asarray(coronal91loc)
coronal92loc = np.asarray(coronal92loc)
coronal93loc = np.asarray(coronal93loc)
coronal94loc = np.asarray(coronal94loc)
coronal95loc = np.asarray(coronal95loc)
coronal96loc = np.asarray(coronal96loc)
coronal97loc = np.asarray(coronal97loc)
coronal98loc = np.asarray(coronal98loc)
coronal99loc = np.asarray(coronal99loc)
coronal90_tensor = prepPNGimgs(coronal90loc)
coronal91_tensor = prepPNGimgs(coronal91loc)
coronal92_tensor = prepPNGimgs(coronal92loc)
coronal93_tensor = prepPNGimgs(coronal93loc)
coronal94_tensor = prepPNGimgs(coronal94loc)
coronal95_tensor = prepPNGimgs(coronal95loc)
coronal96_tensor = prepPNGimgs(coronal96loc)
coronal97_tensor = prepPNGimgs(coronal97loc)
coronal98_tensor = prepPNGimgs(coronal98loc)
coronal99_tensor = prepPNGimgs(coronal99loc)
Y_CDR = np.squeeze(np.asarray(Y_CDR))
Y_CDR_binary = np.asarray(Y_CDR_binary)
Y_CDR_encoded = np.asarray(Y_CDR_encoded)
Y_MMSE = np.squeeze(np.asarray(Y_MMSE))
```

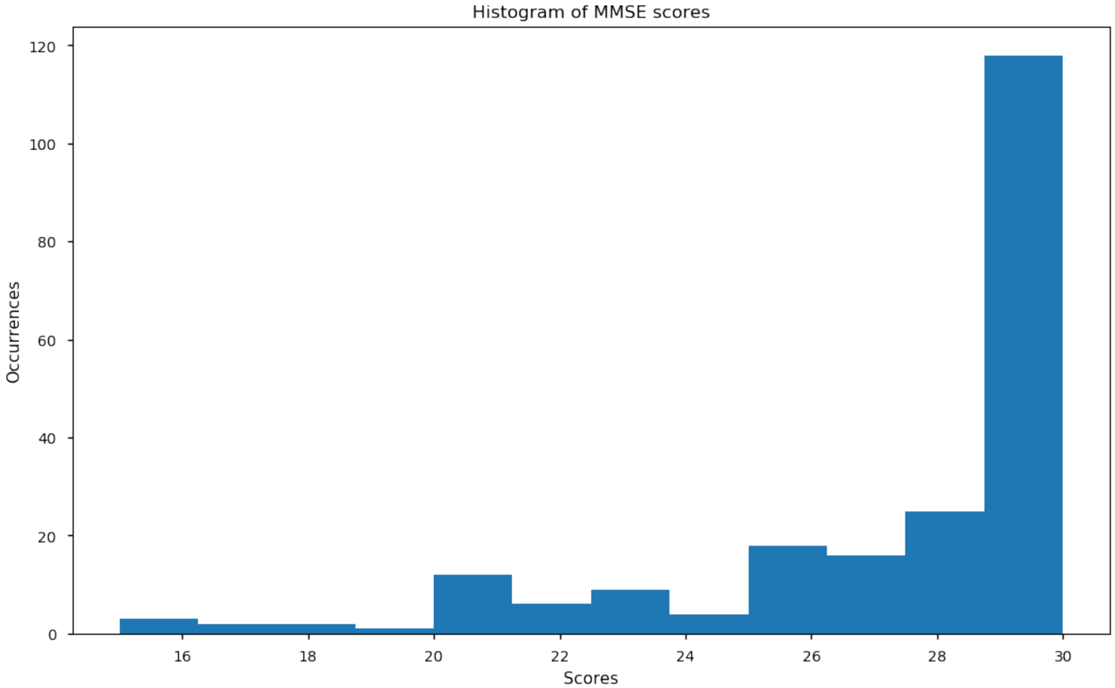
```
Y_MMSE_binary = np.asarray(Y_MMSE_binary)
Y_MMSE_encoded = np.asarray(Y_MMSE_encoded)
```

[10]:
```
plt.hist(X_eTIV)
```

[10]:
```
(array([ 9., 18., 51., 48., 42., 16., 16., 11.,  3.,  2.]),
 array([1122.77 , 1209.663, 1296.556, 1383.449, 1470.342, 1557.235,
        1644.128, 1731.021, 1817.914, 1904.807, 1991.7  ]),
 <a list of 10 Patch objects>)
```

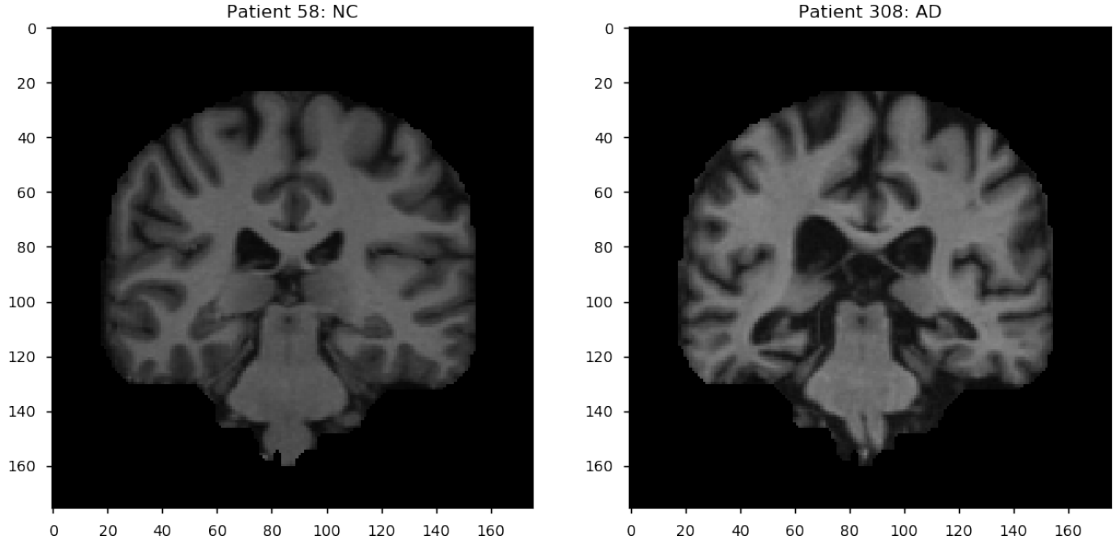



[11]:
```
plt.figure(1, figsize=(12.5, 7.5), dpi=100)

hist = plt.hist(Y_MMSE, bins='auto')
plt.title('Histogram of MMSE scores')
plt.xlabel('Scores')
plt.ylabel('Occurrences')
plt.show()
```



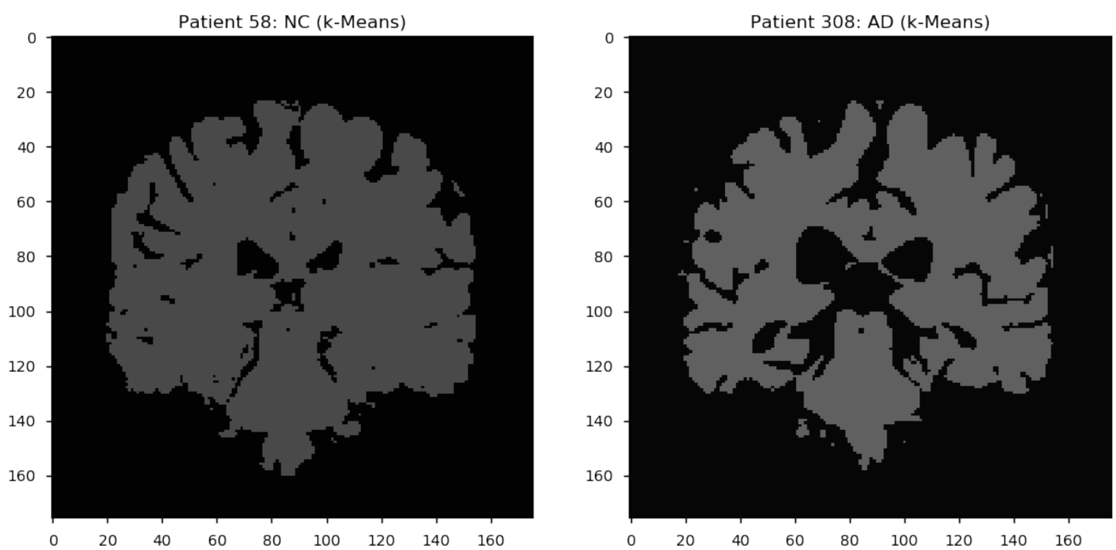

## 3 Analysis

[12]:
```
plt.figure(1, figsize=(12.5, 7.5), dpi=100)
plt.subplot(121)
plt.imshow(cv2.imread("{}".format(coronal90loc[HealthyExampleIndex])))
plt.title("Patient 58: NC")
plt.subplot(122)
plt.title("Patient 308: AD")
plt.imshow(cv2.imread("{}".format(coronal90loc[SickExampleIndex])))
plt.show()
```




Original scans of the example patients.

[13]:
```
plt.figure(2, figsize=(12.5, 7.5), dpi=100)

plt.subplot(121)
plt.imshow(coronal90_tensor[HealthyExampleIndex])
plt.title("Patient 58: NC (k-Means)")

plt.subplot(122)
plt.imshow(coronal90_tensor[SickExampleIndex], cmap='spring')
plt.title("Patient 308: AD (k-Means)")

plt.show()
```



[14]:
```python
patient_holder = np.concatenate((coronal90_tensor,
                                 coronal91_tensor,
                                 coronal92_tensor,
                                 coronal93_tensor,
                                 coronal94_tensor,
                                 coronal95_tensor,
                                 coronal96_tensor,
                                 coronal97_tensor,
                                 coronal98_tensor,
                                 coronal99_tensor))
print(patient_holder.shape)

fin_Y_CDR_encoded = np.tile(Y_CDR_encoded, [10,1])

print(fin_Y_CDR_encoded.shape)
```

```
(2160, 176, 176, 3)
(2160, 2)
```

Scans after post processing.

[15]:
```python
train_percentage_as_decimal = 0.70
end = round(train_percentage_as_decimal*patient_holder.shape[0])
print(end)

X_train_tensor = patient_holder[0:end]
Y_train_output = fin_Y_CDR_encoded[0:end]
X_test_tensor = patient_holder[end:patient_holder.shape[0]]
Y_test_output = fin_Y_CDR_encoded[end:patient_holder.shape[0]]

print(Y_train_output.shape)
print("{}% of the training sample has No Condition".format((100*Y_train_output[:
 ↪,0].sum()/Y_train_output.shape[0])))
print("{}% of the training sample has Alzheimer's Disease".
 ↪format((100*Y_train_output[:,1].sum()/Y_train_output.shape[0])))
print(Y_test_output.shape)
print("{}% of the validation testing sample has No Condition".
 ↪format((100*Y_test_output[:,0].sum()/Y_test_output.shape[0])))
print("{}% of the validation testing sample has Alzheimer's Disease".
 ↪format((100*Y_test_output[:,1].sum()/Y_test_output.shape[0])))
```

```
1512
(1512, 2)
61.574074074074076% of the training sample has No Condition
38.425925925925924% of the training sample has Alzheimer's Disease
(648, 2)
```

```
61.574074074074076% of the validation testing sample has No Condition
38.425925925925924% of the validation testing sample has Alzheimer's Disease
```

[16]:
```
import keras
import tensorflow as tf
from keras import backend as K


print('Keras: ', keras.__version__, 'Tensorflow: ', tf.__version__)
K.tensorflow_backend._get_available_gpus()
```

```
Using TensorFlow backend.

Keras:  2.2.4 Tensorflow:  1.13.1
```

[16]:
```
['/job:localhost/replica:0/task:0/device:GPU:0']
```

[17]:
```
from keras import Sequential
from keras.layers import Convolution2D, ZeroPadding2D, MaxPooling2D, Flatten,␣
 ↪Dropout, Dense

model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(176,176,3)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
```

```python
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, (3, 3), activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(1028, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1028, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['categorical_accuracy'])

model.summary()
```

```
WARNING:tensorflow:From F:\Anaconda\envs\devModeOn\lib\site-
packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From F:\Anaconda\envs\devModeOn\lib\site-
packages\keras\backend\tensorflow_backend.py:3445: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
zero_padding2d_1 (ZeroPaddin (None, 178, 178, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 176, 176, 64)      1792
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 178, 178, 64)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 176, 176, 64)      36928
_________________________________________________________________
```

```
max_pooling2d_1 (MaxPooling2 (None, 88, 88, 64)        0
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 90, 90, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 88, 88, 128)       73856
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 90, 90, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 88, 88, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 44, 44, 128)       0
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 46, 46, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 44, 44, 256)       295168
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 46, 46, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 44, 44, 256)       590080
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 46, 46, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 44, 44, 256)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 22, 22, 256)       0
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 24, 24, 256)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 22, 22, 512)       1180160
_________________________________________________________________
zero_padding2d_9 (ZeroPaddin (None, 24, 24, 512)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 22, 22, 512)       2359808
_________________________________________________________________
zero_padding2d_10 (ZeroPaddi (None, 24, 24, 512)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 22, 22, 512)       2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 11, 11, 512)       0
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 13, 13, 512)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 11, 11, 512)       2359808
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 13, 13, 512)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 11, 11, 512)       2359808
_________________________________________________________________
```

```
zero_padding2d_13 (ZeroPaddi (None, 13, 13, 512)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 11, 11, 512)       2359808
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 5, 5, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12800)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1028)              13159428
_________________________________________________________________
dropout_1 (Dropout)          (None, 1028)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1028)              1057812
_________________________________________________________________
dropout_2 (Dropout)          (None, 1028)              0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 2058
=================================================================
Total params: 28,933,986
Trainable params: 28,933,986
Non-trainable params: 0
_________________________________________________________________
```

[18]:

```python
checkpoint_path="cp.ckpt"
checkpoint_dir=os.path.dirname(checkpoint_path)
cp_callback=tf.keras.callbacks.
 ↪ModelCheckpoint(checkpoint_path,save_weights_only=True,verbose=1)
```

[19]:

```python
normhistory = model.fit(X_train_tensor, Y_train_output,
                        batch_size=32,
                        epochs=150,
                        verbose=1,
                        shuffle=True,
                        validation_data=(X_test_tensor, Y_test_output),
                        callbacks=[cp_callback])



score = model.evaluate(X_test_tensor, Y_test_output)

print('\nTest loss:', score[0])
print('\nTest accuracy:', score[1])
```

```
WARNING:tensorflow:From F:\Anaconda\envs\devModeOn\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
```

```
Train on 1512 samples, validate on 648 samples
Epoch 1/150
1512/1512 [==============================] - 40s 27ms/step - loss: 0.6836 -
categorical_accuracy: 0.6118 - val_loss: 0.6751 - val_categorical_accuracy:
0.6157

Epoch 00001: saving model to cp.ckpt
Epoch 2/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6720 -
categorical_accuracy: 0.6157 - val_loss: 0.6689 - val_categorical_accuracy:
0.6157

Epoch 00002: saving model to cp.ckpt
Epoch 3/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6683 -
categorical_accuracy: 0.6157 - val_loss: 0.6667 - val_categorical_accuracy:
0.6157

Epoch 00003: saving model to cp.ckpt
Epoch 4/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6670 -
categorical_accuracy: 0.6157 - val_loss: 0.6663 - val_categorical_accuracy:
0.6157

Epoch 00004: saving model to cp.ckpt
Epoch 5/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00005: saving model to cp.ckpt
Epoch 6/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6669 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00006: saving model to cp.ckpt
Epoch 7/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00007: saving model to cp.ckpt
Epoch 8/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6667 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157
```

```
Epoch 00008: saving model to cp.ckpt
Epoch 9/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6660 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00009: saving model to cp.ckpt
Epoch 10/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00010: saving model to cp.ckpt
Epoch 11/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6671 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00011: saving model to cp.ckpt
Epoch 12/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00012: saving model to cp.ckpt
Epoch 13/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00013: saving model to cp.ckpt
Epoch 14/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6659 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00014: saving model to cp.ckpt
Epoch 15/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6665 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00015: saving model to cp.ckpt
Epoch 16/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6669 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157
```

```
Epoch 00016: saving model to cp.ckpt
Epoch 17/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00017: saving model to cp.ckpt
Epoch 18/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00018: saving model to cp.ckpt
Epoch 19/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00019: saving model to cp.ckpt
Epoch 20/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00020: saving model to cp.ckpt
Epoch 21/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6655 -
categorical_accuracy: 0.6157 - val_loss: 0.6662 - val_categorical_accuracy:
0.6157

Epoch 00021: saving model to cp.ckpt
Epoch 22/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00022: saving model to cp.ckpt
Epoch 23/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157

Epoch 00023: saving model to cp.ckpt
Epoch 24/150
1512/1512 [==============================] - 22s 15ms/step - loss: 0.6667 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157
```

```
Epoch 00024: saving model to cp.ckpt
Epoch 25/150
1512/1512 [==============================] - 22s 15ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00025: saving model to cp.ckpt
Epoch 26/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6661 - val_categorical_accuracy:
0.6157

Epoch 00026: saving model to cp.ckpt
Epoch 27/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00027: saving model to cp.ckpt
Epoch 28/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6668 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00028: saving model to cp.ckpt
Epoch 29/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157

Epoch 00029: saving model to cp.ckpt
Epoch 30/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157

Epoch 00030: saving model to cp.ckpt
Epoch 31/150
1512/1512 [==============================] - 22s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157

Epoch 00031: saving model to cp.ckpt
Epoch 32/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157
```

```
Epoch 00032: saving model to cp.ckpt
Epoch 33/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6658 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00033: saving model to cp.ckpt
Epoch 34/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6657 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00034: saving model to cp.ckpt
Epoch 35/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6660 -
categorical_accuracy: 0.6157 - val_loss: 0.6659 - val_categorical_accuracy:
0.6157

Epoch 00035: saving model to cp.ckpt
Epoch 36/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6659 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00036: saving model to cp.ckpt
Epoch 37/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6659 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00037: saving model to cp.ckpt
Epoch 38/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6652 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00038: saving model to cp.ckpt
Epoch 39/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6660 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00039: saving model to cp.ckpt
Epoch 40/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6659 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157
```

```
Epoch 00040: saving model to cp.ckpt
Epoch 41/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00041: saving model to cp.ckpt
Epoch 42/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00042: saving model to cp.ckpt
Epoch 43/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6663 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00043: saving model to cp.ckpt
Epoch 44/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6661 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00044: saving model to cp.ckpt
Epoch 45/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6664 -
categorical_accuracy: 0.6157 - val_loss: 0.6658 - val_categorical_accuracy:
0.6157

Epoch 00045: saving model to cp.ckpt
Epoch 46/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6665 -
categorical_accuracy: 0.6157 - val_loss: 0.6656 - val_categorical_accuracy:
0.6157

Epoch 00046: saving model to cp.ckpt
Epoch 47/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6655 - val_categorical_accuracy:
0.6157

Epoch 00047: saving model to cp.ckpt
Epoch 48/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6657 -
categorical_accuracy: 0.6157 - val_loss: 0.6654 - val_categorical_accuracy:
0.6157
```

```
Epoch 00048: saving model to cp.ckpt
Epoch 49/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6660 -
categorical_accuracy: 0.6157 - val_loss: 0.6654 - val_categorical_accuracy:
0.6157

Epoch 00049: saving model to cp.ckpt
Epoch 50/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6656 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00050: saving model to cp.ckpt
Epoch 51/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6653 -
categorical_accuracy: 0.6157 - val_loss: 0.6657 - val_categorical_accuracy:
0.6157

Epoch 00051: saving model to cp.ckpt
Epoch 52/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6652 -
categorical_accuracy: 0.6157 - val_loss: 0.6660 - val_categorical_accuracy:
0.6157

Epoch 00052: saving model to cp.ckpt
Epoch 53/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6651 -
categorical_accuracy: 0.6157 - val_loss: 0.6652 - val_categorical_accuracy:
0.6157

Epoch 00053: saving model to cp.ckpt
Epoch 54/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6656 -
categorical_accuracy: 0.6157 - val_loss: 0.6653 - val_categorical_accuracy:
0.6157

Epoch 00054: saving model to cp.ckpt
Epoch 55/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6652 -
categorical_accuracy: 0.6157 - val_loss: 0.6651 - val_categorical_accuracy:
0.6157

Epoch 00055: saving model to cp.ckpt
Epoch 56/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6646 -
categorical_accuracy: 0.6157 - val_loss: 0.6654 - val_categorical_accuracy:
0.6157
```

```
Epoch 00056: saving model to cp.ckpt
Epoch 57/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6655 -
categorical_accuracy: 0.6157 - val_loss: 0.6648 - val_categorical_accuracy:
0.6157

Epoch 00057: saving model to cp.ckpt
Epoch 58/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6662 -
categorical_accuracy: 0.6157 - val_loss: 0.6649 - val_categorical_accuracy:
0.6157

Epoch 00058: saving model to cp.ckpt
Epoch 59/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6646 -
categorical_accuracy: 0.6157 - val_loss: 0.6647 - val_categorical_accuracy:
0.6157

Epoch 00059: saving model to cp.ckpt
Epoch 60/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6647 -
categorical_accuracy: 0.6157 - val_loss: 0.6645 - val_categorical_accuracy:
0.6157

Epoch 00060: saving model to cp.ckpt
Epoch 61/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6656 -
categorical_accuracy: 0.6157 - val_loss: 0.6644 - val_categorical_accuracy:
0.6157

Epoch 00061: saving model to cp.ckpt
Epoch 62/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6638 -
categorical_accuracy: 0.6157 - val_loss: 0.6641 - val_categorical_accuracy:
0.6157

Epoch 00062: saving model to cp.ckpt
Epoch 63/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6646 -
categorical_accuracy: 0.6157 - val_loss: 0.6647 - val_categorical_accuracy:
0.6157

Epoch 00063: saving model to cp.ckpt
Epoch 64/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6648 -
categorical_accuracy: 0.6157 - val_loss: 0.6641 - val_categorical_accuracy:
0.6157
```

```
Epoch 00064: saving model to cp.ckpt
Epoch 65/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6643 -
categorical_accuracy: 0.6157 - val_loss: 0.6632 - val_categorical_accuracy:
0.6157

Epoch 00065: saving model to cp.ckpt
Epoch 66/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6632 -
categorical_accuracy: 0.6157 - val_loss: 0.6629 - val_categorical_accuracy:
0.6157

Epoch 00066: saving model to cp.ckpt
Epoch 67/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6633 -
categorical_accuracy: 0.6157 - val_loss: 0.6626 - val_categorical_accuracy:
0.6157

Epoch 00067: saving model to cp.ckpt
Epoch 68/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6626 -
categorical_accuracy: 0.6157 - val_loss: 0.6637 - val_categorical_accuracy:
0.6157

Epoch 00068: saving model to cp.ckpt
Epoch 69/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6633 -
categorical_accuracy: 0.6157 - val_loss: 0.6649 - val_categorical_accuracy:
0.6157

Epoch 00069: saving model to cp.ckpt
Epoch 70/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6624 -
categorical_accuracy: 0.6157 - val_loss: 0.6610 - val_categorical_accuracy:
0.6157

Epoch 00070: saving model to cp.ckpt
Epoch 71/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6622 -
categorical_accuracy: 0.6157 - val_loss: 0.6593 - val_categorical_accuracy:
0.6157

Epoch 00071: saving model to cp.ckpt
Epoch 72/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6608 -
categorical_accuracy: 0.6157 - val_loss: 0.6605 - val_categorical_accuracy:
0.6157
```

```
Epoch 00072: saving model to cp.ckpt
Epoch 73/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6592 -
categorical_accuracy: 0.6157 - val_loss: 0.6561 - val_categorical_accuracy:
0.6157

Epoch 00073: saving model to cp.ckpt
Epoch 74/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6613 -
categorical_accuracy: 0.6157 - val_loss: 0.6559 - val_categorical_accuracy:
0.6157

Epoch 00074: saving model to cp.ckpt
Epoch 75/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6585 -
categorical_accuracy: 0.6157 - val_loss: 0.6625 - val_categorical_accuracy:
0.6157

Epoch 00075: saving model to cp.ckpt
Epoch 76/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6579 -
categorical_accuracy: 0.6157 - val_loss: 0.6614 - val_categorical_accuracy:
0.6157

Epoch 00076: saving model to cp.ckpt
Epoch 77/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6524 -
categorical_accuracy: 0.6157 - val_loss: 0.6647 - val_categorical_accuracy:
0.6157

Epoch 00077: saving model to cp.ckpt
Epoch 78/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6510 -
categorical_accuracy: 0.6157 - val_loss: 0.6651 - val_categorical_accuracy:
0.6157

Epoch 00078: saving model to cp.ckpt
Epoch 79/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6534 -
categorical_accuracy: 0.6157 - val_loss: 0.6827 - val_categorical_accuracy:
0.6157

Epoch 00079: saving model to cp.ckpt
Epoch 80/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6494 -
categorical_accuracy: 0.6157 - val_loss: 0.6810 - val_categorical_accuracy:
0.6157
```

```
Epoch 00080: saving model to cp.ckpt
Epoch 81/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6461 -
categorical_accuracy: 0.6184 - val_loss: 0.6274 - val_categorical_accuracy:
0.6157

Epoch 00081: saving model to cp.ckpt
Epoch 82/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6463 -
categorical_accuracy: 0.6164 - val_loss: 0.6795 - val_categorical_accuracy:
0.6157

Epoch 00082: saving model to cp.ckpt
Epoch 83/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6480 -
categorical_accuracy: 0.6184 - val_loss: 0.6407 - val_categorical_accuracy:
0.6157

Epoch 00083: saving model to cp.ckpt
Epoch 84/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6371 -
categorical_accuracy: 0.6171 - val_loss: 0.6277 - val_categorical_accuracy:
0.6373

Epoch 00084: saving model to cp.ckpt
Epoch 85/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6338 -
categorical_accuracy: 0.6263 - val_loss: 0.6225 - val_categorical_accuracy:
0.6157

Epoch 00085: saving model to cp.ckpt
Epoch 86/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6301 -
categorical_accuracy: 0.6362 - val_loss: 0.5929 - val_categorical_accuracy:
0.6867

Epoch 00086: saving model to cp.ckpt
Epoch 87/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6190 -
categorical_accuracy: 0.6448 - val_loss: 0.8329 - val_categorical_accuracy:
0.6157

Epoch 00087: saving model to cp.ckpt
Epoch 88/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6271 -
categorical_accuracy: 0.6462 - val_loss: 0.5782 - val_categorical_accuracy:
0.6759
```

```
Epoch 00088: saving model to cp.ckpt
Epoch 89/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6164 -
categorical_accuracy: 0.6396 - val_loss: 0.6650 - val_categorical_accuracy:
0.6157

Epoch 00089: saving model to cp.ckpt
Epoch 90/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.6088 -
categorical_accuracy: 0.6415 - val_loss: 0.5719 - val_categorical_accuracy:
0.7160

Epoch 00090: saving model to cp.ckpt
Epoch 91/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5964 -
categorical_accuracy: 0.6845 - val_loss: 0.7184 - val_categorical_accuracy:
0.6204

Epoch 00091: saving model to cp.ckpt
Epoch 92/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5913 -
categorical_accuracy: 0.6792 - val_loss: 0.5459 - val_categorical_accuracy:
0.6821

Epoch 00092: saving model to cp.ckpt
Epoch 93/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5770 -
categorical_accuracy: 0.6806 - val_loss: 0.5455 - val_categorical_accuracy:
0.6836

Epoch 00093: saving model to cp.ckpt
Epoch 94/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5662 -
categorical_accuracy: 0.6964 - val_loss: 0.5310 - val_categorical_accuracy:
0.7531

Epoch 00094: saving model to cp.ckpt
Epoch 95/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5604 -
categorical_accuracy: 0.7024 - val_loss: 0.6195 - val_categorical_accuracy:
0.6790

Epoch 00095: saving model to cp.ckpt
Epoch 96/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5477 -
categorical_accuracy: 0.7103 - val_loss: 0.6043 - val_categorical_accuracy:
0.6867
```

```
Epoch 00096: saving model to cp.ckpt
Epoch 97/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5344 -
categorical_accuracy: 0.7255 - val_loss: 0.6432 - val_categorical_accuracy:
0.6836

Epoch 00097: saving model to cp.ckpt
Epoch 98/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5398 -
categorical_accuracy: 0.7156 - val_loss: 0.5121 - val_categorical_accuracy:
0.7330

Epoch 00098: saving model to cp.ckpt
Epoch 99/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5389 -
categorical_accuracy: 0.7235 - val_loss: 0.5526 - val_categorical_accuracy:
0.7006

Epoch 00099: saving model to cp.ckpt
Epoch 100/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5158 -
categorical_accuracy: 0.7235 - val_loss: 0.5074 - val_categorical_accuracy:
0.7531

Epoch 00100: saving model to cp.ckpt
Epoch 101/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5161 -
categorical_accuracy: 0.7288 - val_loss: 0.4902 - val_categorical_accuracy:
0.7515

Epoch 00101: saving model to cp.ckpt
Epoch 102/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5072 -
categorical_accuracy: 0.7354 - val_loss: 0.5337 - val_categorical_accuracy:
0.7207

Epoch 00102: saving model to cp.ckpt
Epoch 103/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.5006 -
categorical_accuracy: 0.7388 - val_loss: 0.5851 - val_categorical_accuracy:
0.6759

Epoch 00103: saving model to cp.ckpt
Epoch 104/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4930 -
categorical_accuracy: 0.7368 - val_loss: 0.4957 - val_categorical_accuracy:
0.7546
```

```
Epoch 00104: saving model to cp.ckpt
Epoch 105/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4957 -
categorical_accuracy: 0.7434 - val_loss: 0.5067 - val_categorical_accuracy:
0.7238

Epoch 00105: saving model to cp.ckpt
Epoch 106/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4790 -
categorical_accuracy: 0.7487 - val_loss: 0.5994 - val_categorical_accuracy:
0.7160

Epoch 00106: saving model to cp.ckpt
Epoch 107/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4769 -
categorical_accuracy: 0.7467 - val_loss: 0.6056 - val_categorical_accuracy:
0.7145

Epoch 00107: saving model to cp.ckpt
Epoch 108/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4703 -
categorical_accuracy: 0.7672 - val_loss: 0.4883 - val_categorical_accuracy:
0.7330

Epoch 00108: saving model to cp.ckpt
Epoch 109/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4582 -
categorical_accuracy: 0.7665 - val_loss: 0.5743 - val_categorical_accuracy:
0.6404

Epoch 00109: saving model to cp.ckpt
Epoch 110/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4559 -
categorical_accuracy: 0.7639 - val_loss: 0.5799 - val_categorical_accuracy:
0.7037

Epoch 00110: saving model to cp.ckpt
Epoch 111/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4430 -
categorical_accuracy: 0.7884 - val_loss: 0.6282 - val_categorical_accuracy:
0.6713

Epoch 00111: saving model to cp.ckpt
Epoch 112/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4434 -
categorical_accuracy: 0.7771 - val_loss: 0.5822 - val_categorical_accuracy:
0.6991
```

```
Epoch 00112: saving model to cp.ckpt
Epoch 113/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4263 -
categorical_accuracy: 0.7897 - val_loss: 0.4788 - val_categorical_accuracy:
0.7608

Epoch 00113: saving model to cp.ckpt
Epoch 114/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4295 -
categorical_accuracy: 0.7817 - val_loss: 0.5886 - val_categorical_accuracy:
0.6867

Epoch 00114: saving model to cp.ckpt
Epoch 115/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.4070 -
categorical_accuracy: 0.7917 - val_loss: 0.8890 - val_categorical_accuracy:
0.7346

Epoch 00115: saving model to cp.ckpt
Epoch 116/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3923 -
categorical_accuracy: 0.8135 - val_loss: 0.4919 - val_categorical_accuracy:
0.7485

Epoch 00116: saving model to cp.ckpt
Epoch 117/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3947 -
categorical_accuracy: 0.8175 - val_loss: 0.4809 - val_categorical_accuracy:
0.7701

Epoch 00117: saving model to cp.ckpt
Epoch 118/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3639 -
categorical_accuracy: 0.8234 - val_loss: 0.4884 - val_categorical_accuracy:
0.7562

Epoch 00118: saving model to cp.ckpt
Epoch 119/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3557 -
categorical_accuracy: 0.8366 - val_loss: 0.5171 - val_categorical_accuracy:
0.7685

Epoch 00119: saving model to cp.ckpt
Epoch 120/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3369 -
categorical_accuracy: 0.8426 - val_loss: 0.6539 - val_categorical_accuracy:
0.7469
```

```
Epoch 00120: saving model to cp.ckpt
Epoch 121/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3248 -
categorical_accuracy: 0.8452 - val_loss: 0.4964 - val_categorical_accuracy:
0.7500

Epoch 00121: saving model to cp.ckpt
Epoch 122/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3045 -
categorical_accuracy: 0.8697 - val_loss: 1.2722 - val_categorical_accuracy:
0.5772

Epoch 00122: saving model to cp.ckpt
Epoch 123/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3205 -
categorical_accuracy: 0.8677 - val_loss: 0.4696 - val_categorical_accuracy:
0.7731

Epoch 00123: saving model to cp.ckpt
Epoch 124/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.2493 -
categorical_accuracy: 0.8862 - val_loss: 0.6429 - val_categorical_accuracy:
0.6975

Epoch 00124: saving model to cp.ckpt
Epoch 125/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.2469 -
categorical_accuracy: 0.8962 - val_loss: 0.5800 - val_categorical_accuracy:
0.6991

Epoch 00125: saving model to cp.ckpt
Epoch 126/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.2413 -
categorical_accuracy: 0.9001 - val_loss: 0.4307 - val_categorical_accuracy:
0.8102

Epoch 00126: saving model to cp.ckpt
Epoch 127/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.2312 -
categorical_accuracy: 0.9054 - val_loss: 0.5510 - val_categorical_accuracy:
0.7994

Epoch 00127: saving model to cp.ckpt
Epoch 128/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.1595 -
categorical_accuracy: 0.9378 - val_loss: 0.5624 - val_categorical_accuracy:
0.7870
```

```
Epoch 00128: saving model to cp.ckpt
Epoch 129/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.1470 -
categorical_accuracy: 0.9484 - val_loss: 0.4558 - val_categorical_accuracy:
0.8333

Epoch 00129: saving model to cp.ckpt
Epoch 130/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.1474 -
categorical_accuracy: 0.9444 - val_loss: 0.5281 - val_categorical_accuracy:
0.8071

Epoch 00130: saving model to cp.ckpt
Epoch 131/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0922 -
categorical_accuracy: 0.9636 - val_loss: 0.4525 - val_categorical_accuracy:
0.8272

Epoch 00131: saving model to cp.ckpt
Epoch 132/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0620 -
categorical_accuracy: 0.9835 - val_loss: 0.4952 - val_categorical_accuracy:
0.8133

Epoch 00132: saving model to cp.ckpt
Epoch 133/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.1263 -
categorical_accuracy: 0.9484 - val_loss: 0.5890 - val_categorical_accuracy:
0.8025

Epoch 00133: saving model to cp.ckpt
Epoch 134/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0312 -
categorical_accuracy: 0.9940 - val_loss: 0.4840 - val_categorical_accuracy:
0.8302

Epoch 00134: saving model to cp.ckpt
Epoch 135/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0178 -
categorical_accuracy: 0.9954 - val_loss: 0.7237 - val_categorical_accuracy:
0.8117

Epoch 00135: saving model to cp.ckpt
Epoch 136/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0032 -
categorical_accuracy: 1.0000 - val_loss: 0.6353 - val_categorical_accuracy:
0.8395
```

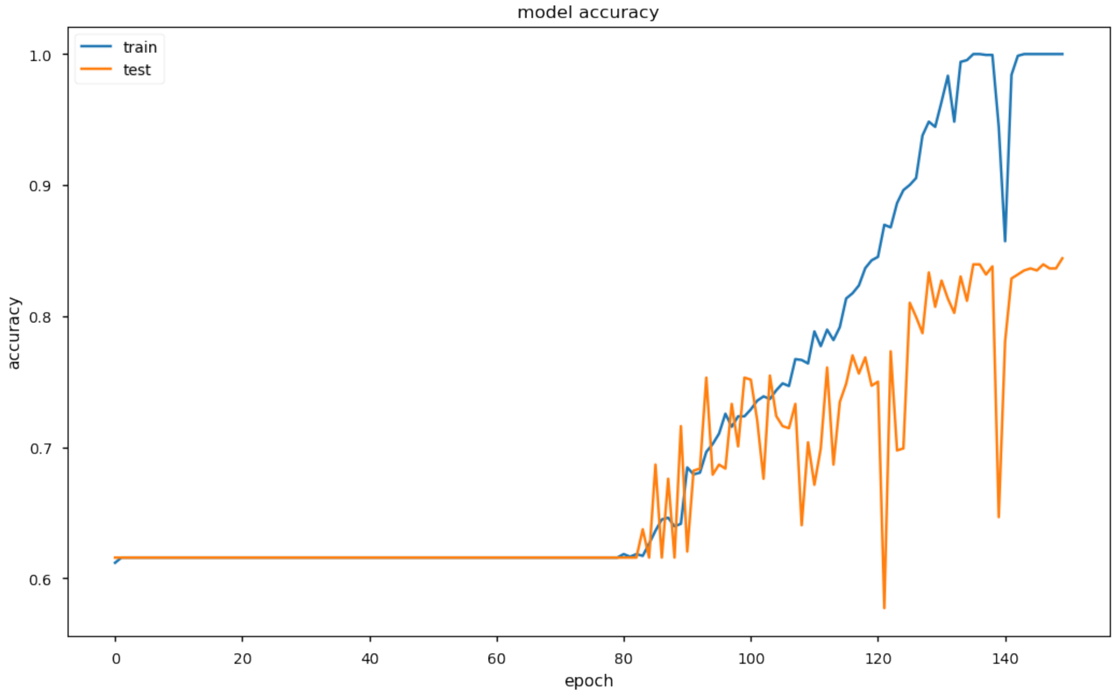
```
Epoch 00136: saving model to cp.ckpt
Epoch 137/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0040 -
categorical_accuracy: 1.0000 - val_loss: 0.6362 - val_categorical_accuracy:
0.8395

Epoch 00137: saving model to cp.ckpt
Epoch 138/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0030 -
categorical_accuracy: 0.9993 - val_loss: 0.8239 - val_categorical_accuracy:
0.8318

Epoch 00138: saving model to cp.ckpt
Epoch 139/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0037 -
categorical_accuracy: 0.9993 - val_loss: 0.7781 - val_categorical_accuracy:
0.8380

Epoch 00139: saving model to cp.ckpt
Epoch 140/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.1287 -
categorical_accuracy: 0.9444 - val_loss: 0.5963 - val_categorical_accuracy:
0.6466

Epoch 00140: saving model to cp.ckpt
Epoch 141/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.3094 -
categorical_accuracy: 0.8571 - val_loss: 0.5087 - val_categorical_accuracy:
0.7809

Epoch 00141: saving model to cp.ckpt
Epoch 142/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0495 -
categorical_accuracy: 0.9841 - val_loss: 0.5424 - val_categorical_accuracy:
0.8287

Epoch 00142: saving model to cp.ckpt
Epoch 143/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0096 -
categorical_accuracy: 0.9987 - val_loss: 0.5966 - val_categorical_accuracy:
0.8318

Epoch 00143: saving model to cp.ckpt
Epoch 144/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0019 -
categorical_accuracy: 1.0000 - val_loss: 0.6825 - val_categorical_accuracy:
0.8349
```

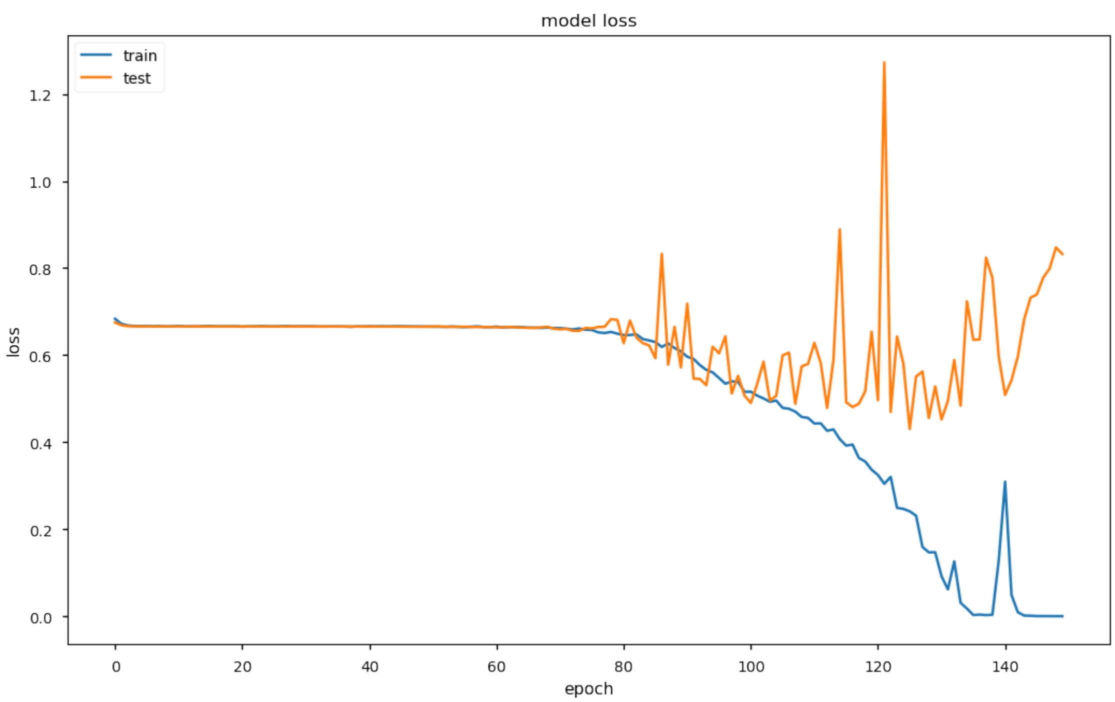
```
Epoch 00144: saving model to cp.ckpt
Epoch 145/150
1512/1512 [==============================] - 21s 14ms/step - loss: 0.0014 -
categorical_accuracy: 1.0000 - val_loss: 0.7321 - val_categorical_accuracy:
0.8364

Epoch 00145: saving model to cp.ckpt
Epoch 146/150
1512/1512 [==============================] - 21s 14ms/step - loss: 7.0639e-04 -
categorical_accuracy: 1.0000 - val_loss: 0.7396 - val_categorical_accuracy:
0.8349

Epoch 00146: saving model to cp.ckpt
Epoch 147/150
1512/1512 [==============================] - 21s 14ms/step - loss: 6.8576e-04 -
categorical_accuracy: 1.0000 - val_loss: 0.7782 - val_categorical_accuracy:
0.8395

Epoch 00147: saving model to cp.ckpt
Epoch 148/150
1512/1512 [==============================] - 21s 14ms/step - loss: 5.2251e-04 -
categorical_accuracy: 1.0000 - val_loss: 0.7990 - val_categorical_accuracy:
0.8364

Epoch 00148: saving model to cp.ckpt
Epoch 149/150
1512/1512 [==============================] - 21s 14ms/step - loss: 4.6317e-04 -
categorical_accuracy: 1.0000 - val_loss: 0.8474 - val_categorical_accuracy:
0.8364

Epoch 00149: saving model to cp.ckpt
Epoch 150/150
1512/1512 [==============================] - 21s 14ms/step - loss: 3.9970e-04 -
categorical_accuracy: 1.0000 - val_loss: 0.8324 - val_categorical_accuracy:
0.8441

Epoch 00150: saving model to cp.ckpt
648/648 [==============================] - 3s 5ms/step

Test loss: 0.8324069212432261

Test accuracy: 0.8441358024691358
```

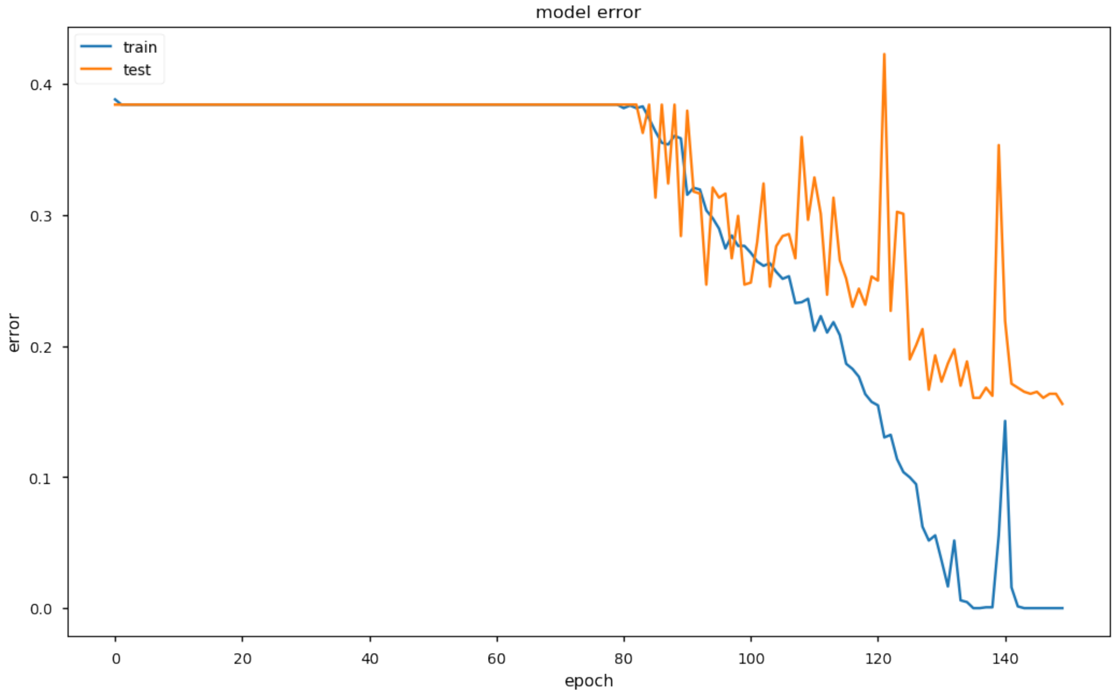
[20]:
```
# list all data in history
print(normhistory.history.keys())
plt.figure(3, figsize=(12.5, 7.5), dpi=100)
# summarize history for accuracy
```

```python
caterror = [1-x for x in normhistory.history['categorical_accuracy']]
valcaterror = [1-x for x in normhistory.history['val_categorical_accuracy']]
plt.plot(normhistory.history['categorical_accuracy'])
plt.plot(normhistory.history['val_categorical_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.figure(3, figsize=(12.5, 7.5), dpi=100)
plt.plot(normhistory.history['loss'])
plt.plot(normhistory.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.figure(3, figsize=(12.5, 7.5), dpi=100)
plt.plot(caterror)
plt.plot(valcaterror)
plt.title('model error')
plt.ylabel('error')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['val_loss', 'val_categorical_accuracy', 'loss',
'categorical_accuracy'])
```


model accuracy
train
test
1.0
0.9
0.8
0.7
0.6
accuracy
0
20
40
60
80
100
120
140
epoch


model loss
train
test
1.2
1.0
0.8
0.6
0.4
0.2
0.0
loss
0
20
40
60
80
100
120
140
epoch




[21]:
```
print(caterror.index(min(caterror)))
print(valcaterror.index(min(valcaterror)))
print(np.std(caterror))
print(np.mean(caterror))
print(np.std(valcaterror))
print(np.mean(valcaterror))
```

```
135
149
0.13619149968302421
0.287936507936508
0.08011994281281755
0.3263991769547325
```