Project – Ocular Disease Image Classification (work in progress)

October 31, 2020

[3]:
```
# import important dependencies
import random, os, re, math, itertools

# data manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('seaborn-whitegrid')
%matplotlib inline

# deep learning
import tensorflow as tf
from tensorflow import keras
import tensorflow_datasets as tfds
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Conv2D, Input, Activation, Reshape,␣
 ↪Flatten, MaxPool2D, GlobalMaxPooling2D, GlobalAveragePooling2D, Concatenate,␣
 ↪Dropout, BatchNormalization
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.optimizers import Adam, SGD, Adadelta
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪img_to_array, load_img

# image manipulation
import cv2
from PIL import Image

# file operations
import glob, shutil

# machine learning
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.metrics import confusion_matrix
```

```
import warnings
warnings.simplefilter(action = 'ignore', category = FutureWarning)
```

0.0.1 Looking at the Report Data

[28]:
```
eye = pd.read_csv('/Users/armaanvalvi/Documents/Ocular Disease Image␣
↪Classification/full_df.csv')
```

[29]:
```
len(eye.filename.unique())
```

[29]: 6392

[30]:
```
eye.columns = ['id', 'age', 'sex', 'left', 'right', 'l-diagnosis',␣
↪'r-diagnosis', 'normal', 'diabetes', 'glaucoma',
               'cataract', 'amd', 'hypertension', 'myopia', 'other', 'filepath',␣
↪'labels', 'target', 'file_name']
```

[31]:
```
eye.drop(['filepath'], axis = 1, inplace = True)
```

[32]:
```
eye.loc[eye['sex'] == 'Male', 'sex'] = 'M'
eye.loc[eye['sex'] == 'Female', 'sex'] = 'F'
# eye2.rename(columns = {'hypertension':'h-tension'})
eye.head(2)
```

[32]:

	id	age	sex	left	right	l-diagnosis	r-diagnosis	normal	diabetes	glaucoma	cataract	amd	hypertension	myopia	other	labels	target	file_name
0	0	69	F	0_left.jpg	0_right.jpg	cataract	normal fundus	0	0	0	1	0	0	0	0	['N']	[1, 0, 0, 0, 0, 0, 0, 0]	0_right.jpg
1	1	57	M	1_left.jpg	1_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	0	['N']	[1, 0, 0, 0, 0, 0, 0, 0]	1_right.jpg

[33]:
```
# rearranging the columns of the data frame
list_columns = eye.columns.tolist()
eye_final = eye[[list_columns[-1]] + list_columns[0:5] + list_columns[15:17] +␣
↪list_columns[5:15]]
eye_final.head(2)
```

[33]:

	file_name	id	age	sex	left	right	labels	diabetes	glaucoma	cataract	amd	hypertension	myopia	other	target
0	0_right.jpg	0	69	F	0_left.jpg	0_right.jpg	['N']	0	0	1	0	0	0	0	[1, 0, 0, 0, 0, 0, 0, 0]
1	1_right.jpg	1	57	M	1_left.jpg	1_right.jpg	['N']	0	0	0	0	0	0	0	[1, 0, 0, 0, 0, 0, 0, 0]

```
                     target    l-diagnosis    r-diagnosis  normal  diabetes  \
0  [1, 0, 0, 0, 0, 0, 0, 0]       cataract  normal fundus       0         0
1  [1, 0, 0, 0, 0, 0, 0, 0]  normal fundus  normal fundus       1         0

   glaucoma  cataract  amd  hypertension  myopia  other
0         0         1    0             0       0      0
1         0         0    0             0       0      0
```

[34]:
```
# making visual edits to the labels column

eye_final.loc[eye_final['labels'] == "['N']", 'labels'] = 'N'
eye_final.loc[eye_final['labels'] == "['D']", 'labels'] = 'D'
eye_final.loc[eye_final['labels'] == "['G']", 'labels'] = 'G'
eye_final.loc[eye_final['labels'] == "['C']", 'labels'] = 'C'
eye_final.loc[eye_final['labels'] == "['A']", 'labels'] = 'A'
eye_final.loc[eye_final['labels'] == "['H']", 'labels'] = 'H'
eye_final.loc[eye_final['labels'] == "['M']", 'labels'] = 'M'
eye_final.loc[eye_final['labels'] == "['O']", 'labels'] = 'O'
```

[35]:
```
# retriving images

files = glob.glob('/Users/armaanvalvi/Documents/project-data/archive/images/*.
 ↪jpg')
data = []
for file in files:
#     img = Image.open(file)
    data.append(file)
```

[36]:
```
# creating a column with the file paths as a data frame
data_df = pd.DataFrame({"file_path":data})
```

[37]:
```
# concatinating the dataframes
eye_final = pd.concat([eye_final, data_df], axis = 1)
```

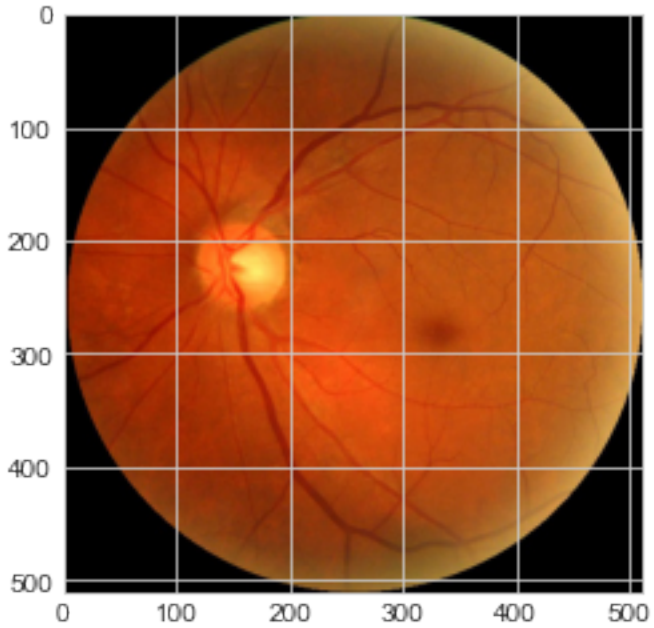
[38]:
```
eye_final.head(2)
```

[38]:
```
     file_name  id  age sex        left        right labels  \
0  0_right.jpg   0   69   F  0_left.jpg  0_right.jpg      N
1  1_right.jpg   1   57   M  1_left.jpg  1_right.jpg      N

                     target    l-diagnosis    r-diagnosis  normal  diabetes  \
0  [1, 0, 0, 0, 0, 0, 0, 0]       cataract  normal fundus       0         0
1  [1, 0, 0, 0, 0, 0, 0, 0]  normal fundus  normal fundus       1         0

   glaucoma  cataract  amd  hypertension  myopia  other  \
0         0         1    0             0       0      0
1         0         0    0             0       0      0
```

```
                                                file_path
0  /Users/armaanvalvi/Documents/project-data/arch…
1  /Users/armaanvalvi/Documents/project-data/arch…
```

[39]:
```
# viewing sample image

pilot_img = Image.open(eye_final['file_path'][0])
plt.imshow(pilot_img)

pilot_img.size
```

[39]: (512, 512)



0.0.2 Basic EDA

[40]:
```
eye_final.groupby('labels').sum()
```

[40]:

labels	id	age	normal	diabetes	glaucoma	cataract	amd
A	269524	16282	0	11	17	0	266
C	552002	19562	0	42	5	293	0
D	4993483	89565	0	1608	18	32	18
G	383847	17898	0	34	284	1	2
H	192546	7302	0	29	6	4	4
M	317821	12734	0	12	0	0	0

```
N       7320912  164283    2101       252        42        51   22
O        487061   42202       0       135        25        21    7

        hypertension  myopia  other
labels
A                  4       3     16
C                  0       0     31
D                 60      19    304
G                  9      11     46
H                128       0     16
M                  0     232     41
N                  0      26    429
O                  2      15    705
```

[41]:
```
eye_final.describe()
```

[41]:
```
                id          age       normal     diabetes     glaucoma  \
count  6392.000000  6392.000000  6392.000000  6392.000000  6392.000000
mean   2271.150814    57.857947     0.328692     0.332134     0.062109
std    1417.559018    11.727737     0.469775     0.471016     0.241372
min       0.000000     1.000000     0.000000     0.000000     0.000000
25%     920.750000    51.000000     0.000000     0.000000     0.000000
50%    2419.500000    59.000000     0.000000     0.000000     0.000000
75%    3294.000000    66.000000     1.000000     1.000000     0.000000
max    4784.000000    91.000000     1.000000     1.000000     1.000000

          cataract          amd  hypertension       myopia        other
count  6392.000000  6392.000000   6392.000000  6392.000000  6392.000000
mean      0.062891     0.049906      0.031758     0.047872     0.248436
std       0.242786     0.217768      0.175370     0.213513     0.432139
min       0.000000     0.000000      0.000000     0.000000     0.000000
25%       0.000000     0.000000      0.000000     0.000000     0.000000
50%       0.000000     0.000000      0.000000     0.000000     0.000000
75%       0.000000     0.000000      0.000000     0.000000     0.000000
max       1.000000     1.000000      1.000000     1.000000     1.000000
```

[42]:
```
eye_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6392 entries, 0 to 6391
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   file_name     6392 non-null   object
 1   id            6392 non-null   int64
 2   age           6392 non-null   int64
 3   sex           6392 non-null   object
```

```
 4   left           6392 non-null   object
 5   right          6392 non-null   object
 6   labels         6392 non-null   object
 7   target         6392 non-null   object
 8   l-diagnosis    6392 non-null   object
 9   r-diagnosis    6392 non-null   object
 10  normal         6392 non-null   int64
 11  diabetes       6392 non-null   int64
 12  glaucoma       6392 non-null   int64
 13  cataract       6392 non-null   int64
 14  amd            6392 non-null   int64
 15  hypertension   6392 non-null   int64
 16  myopia         6392 non-null   int64
 17  other          6392 non-null   int64
 18  file_path      6392 non-null   object
dtypes: int64(10), object(9)
memory usage: 948.9+ KB
```

[43]:
```
eye_final['labels'].value_counts()
```

[43]:
```
N    2873
D    1608
O     708
C     293
G     284
A     266
M     232
H     128
Name: labels, dtype: int64
```

[44]:
```
train_data, test_data = train_test_split(eye_final, test_size = 0.18,␣
↪random_state = 1234)
```

[45]:
```
train_data['labels'].value_counts(), test_data['labels'].value_counts()
```

[45]:
```
(N    2351
 D    1323
 O     598
 C     235
 G     229
 A     212
 M     189
 H     104
 Name: labels, dtype: int64,
 N    522
 D    285
 O    110
```

```
C     58
G     55
A     54
M     43
H     24
Name: labels, dtype: int64)
```

[46]:
```
train_data.loc[train_data['sex'] == 'M', 'sex'] = '0'
train_data.loc[train_data['sex'] == 'F', 'sex'] = '1'
train_data.drop(['right'], axis = 1, inplace = True)
```

[47]:
```
train_data.loc[train_data['file_name'].str.contains('left', regex = True),␣
↪'left'] = '1'
train_data.loc[train_data['file_name'].str.contains('right', regex = True),␣
↪'left'] = '0'
train_data.head(2)
```

[47]:

	file_name	id	age	sex	left	labels	target	\
1941	2799_right.jpg	2799	53	0	0	N	[1, 0, 0, 0, 0, 0, 0, 0]	
6391	4784_left.jpg	4784	58	0	1	H	[0, 0, 0, 0, 0, 1, 0, 0]	

	l-diagnosis	\
1941	normal fundus	
6391	hypertensive retinopathy age-related macular d…	

	r-diagnosis	normal	diabetes	\
1941	normal fundus	1	0	
6391	hypertensive retinopathy age-related macular d…	0	0	

	glaucoma	cataract	amd	hypertension	myopia	other	\
1941	0	0	0	0	0	0	
6391	0	0	1	1	0	0	

	file_path
1941	/Users/armaanvalvi/Documents/project-data/arch…
6391	/Users/armaanvalvi/Documents/project-data/arch…

[48]:
```
train_final = pd.DataFrame(train_data, columns =['sex', 'left'])
one_hot_columns = train_final.columns.tolist()
train_final_enc = train_final.apply(LabelEncoder().fit_transform)
train_diagnosis = pd.DataFrame(train_data, columns = ['age', 'normal',
                                                      'diabetes', 'glaucoma',
                                                      'cataract', 'amd',
                                                      'hypertension', 'myopia',
                                                      'other'
                                                     ])
train_final_enc = pd.concat([train_final_enc, train_diagnosis], axis = 1)
```

[49]:
```
train_final_enc
X_train = train_final_enc
y_train = train_data['target']
```

[50]:
```
len(test_data)
```

[50]:
```
1151
```

0.1 Creating more pictures to create a more balanced image dataset

[51]:
```
"""create lists of all the file_names for each of the different conditions and␣
 ↪keep them ready"""

n_names = list(eye_final['file_name'][eye_final['labels'] == 'N'])
d_names = list(eye_final['file_name'][eye_final['labels'] == 'D'])
o_names = list(eye_final['file_name'][eye_final['labels'] == 'O'])
c_names = list(eye_final['file_name'][eye_final['labels'] == 'C'])
g_names = list(eye_final['file_name'][eye_final['labels'] == 'G'])
a_names = list(eye_final['file_name'][eye_final['labels'] == 'A'])
m_names = list(eye_final['file_name'][eye_final['labels'] == 'M'])
h_names = list(eye_final['file_name'][eye_final['labels'] == 'H'])


n_paths = list(eye_final['file_path'][eye_final['labels'] == 'N'])
d_paths = list(eye_final['file_path'][eye_final['labels'] == 'D'])
o_paths = list(eye_final['file_path'][eye_final['labels'] == 'O'])
c_paths = list(eye_final['file_path'][eye_final['labels'] == 'C'])
g_paths = list(eye_final['file_path'][eye_final['labels'] == 'G'])
a_paths = list(eye_final['file_path'][eye_final['labels'] == 'A'])
m_paths = list(eye_final['file_path'][eye_final['labels'] == 'M'])
h_paths = list(eye_final['file_path'][eye_final['labels'] == 'H'])
```

[52]:
```
"""creating a function that copies images by category to new folders for data␣
 ↪augmentation
"""


def copy_to_folder(paths, category):
    for file in glob.glob('/Users/armaanvalvi/Documents/project-data/archive/
 ↪images/*.jpg'):
        if file in paths:
            shutil.copy(file, f'/Users/armaanvalvi/Documents/project-data/
 ↪archive/images_by_category/{category}')
```

[31]:
```
# copy_to_folder(a_paths, 'A')
# copy_to_folder(c_paths, 'C')
# copy_to_folder(d_paths, 'D')
```

```
# copy_to_folder(g_paths, 'G')
# copy_to_folder(h_paths, 'H')
# copy_to_folder(m_paths, 'M')
# copy_to_folder(n_paths, 'N')
# copy_to_folder(o_paths, 'O')
```

[3]:
```
""" taken from the keras blog site by founder Francois Chollet """

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

[54]:
```
""" creating a function using the datagen object for each of the image␣
↪categories using code from Keras"""

def get_more_images(main, category, main_input, category_input, prefix_input,␣
↪batch_size_input = 10, loop_input = 6):
    for file in glob.glob(f'/Users/armaanvalvi/Documents/project-data/archive/
↪{main}/{category}/*.jpg'):
        img = load_img(file)
        x = img_to_array(img)
        x = x.reshape((1,) + x.shape)
        i = 0
        for batch in datagen.flow(x, batch_size = batch_size_input,
                                  save_to_dir = f'/Users/armaanvalvi/Documents/
↪project-data/archive/{main_input}/{category_input}',
                                  save_prefix = f'{prefix_input}_aug',
                                  save_format = 'jpg'):
            i += 1
            if i >= loop_input:
                break
```

[55]:
```
# get_more_images_2('H', 'H_pilot', 'H', 1, 2)
# get_more_images('train_final', 'A', 'train_final', 'A', 'A')
# get_more_images('train_final', 'C', 'train_final', 'C', 'C')
# get_more_images('train_final', 'D', 'train_final', 'D', 'D')
# get_more_images('train_final', 'G', 'train_final', 'G', 'G')
# get_more_images('train_final', 'H', 'train_final', 'H', 'H')
# get_more_images('train_final', 'M', 'train_final', 'M', 'M')
# get_more_images('train_final', 'N', 'train_final', 'N', 'N')
# get_more_images('train_final', 'O', 'train_final', 'O', 'O')
```

```
# old version of get_more_images

# get_more_images_2('C', 'C', 'C')
# get_more_images_2('D', 'D', 'D', 1, 2)
# get_more_images_2('G', 'G', 'G')
# get_more_images_2('H', 'H', 'H', 5, 20)
# get_more_images_2('M', 'M', 'M')
# get_more_images_2('N', 'N', 'N', 1, 1)
# get_more_images_2('O', 'O', 'O', 2, 4)
```

[56]:
```
eye_final['labels'].value_counts()
```

[56]:
```
N    2873
D    1608
O     708
C     293
G     284
A     266
M     232
H     128
Name: labels, dtype: int64
```

[17]:
```
# randomly moving images to train_set
def move_from(main, category, main_input, category_input, number):
    for pic in random.sample(glob.glob(f'{main}/{category}/*.jpg'), number):
        shutil.move(pic, f'{main_input}/{category_input}')

# randomly copying images to train_set
def copy_from(main, category, main_input, category_input, number):
    for pic in random.sample(glob.glob(f'{main}/{category}/*.jpg'), number):
        shutil.copy(pic, f'{main_input}/{category_input}')
```

[61]:
```
# os.rename('/Users/armaanvalvi/Documents/project-data/archive/train_set',
#           '/Users/armaanvalvi/Documents/project-data/archive/repository')
```

[62]:
```
os.chdir('/Users/armaanvalvi/Documents/project-data/archive')
```

0.1.1 Creating pilot train and valid data to check the running of the models

[66]:
```
if os.path.isdir('pilot_folder/A') is False:
    os.makedirs('pilot_folder/A')
    os.makedirs('pilot_folder/C')
    os.makedirs('pilot_folder/D')
    os.makedirs('pilot_folder/G')
    os.makedirs('pilot_folder/H')
    os.makedirs('pilot_folder/M')
```

```
    os.makedirs('pilot_folder/N')
    os.makedirs('pilot_folder/O')

if os.path.isdir('pilot_valid/A') is False:
    os.makedirs('pilot_valid/A')
    os.makedirs('pilot_valid/C')
    os.makedirs('pilot_valid/D')
    os.makedirs('pilot_valid/G')
    os.makedirs('pilot_valid/H')
    os.makedirs('pilot_valid/M')
    os.makedirs('pilot_valid/N')
    os.makedirs('pilot_valid/O')
```

[221]:
```
# training
# copy_from('images_by_category','A', 'pilot_folder', 'A', 80)
# copy_from('images_by_category','C', 'pilot_folder', 'C', 80)
# copy_from('images_by_category','D', 'pilot_folder', 'D', 80)
# copy_from('images_by_category','G', 'pilot_folder', 'G', 80)
# copy_from('images_by_category','H', 'pilot_folder', 'H', 80)
# copy_from('images_by_category','M', 'pilot_folder', 'M', 80)
# copy_from('images_by_category','N', 'pilot_folder', 'N', 80)
# copy_from('images_by_category','O', 'pilot_folder', 'O', 80)

# get_more_images('train_final', 'A', 'train_final', 'A', 'A')
# get_more_images('train_final', 'C', 'train_final', 'C', 'C')
# get_more_images('train_final', 'D', 'train_final', 'D', 'D')
# get_more_images('train_final', 'G', 'train_final', 'G', 'G')
# get_more_images('train_final', 'H', 'train_final', 'H', 'H')
# get_more_images('train_final', 'M', 'train_final', 'M', 'M')
# get_more_images('train_final', 'N', 'train_final', 'N', 'N')
# get_more_images('train_final', 'O', 'train_final', 'O', 'O')
```

[222]:
```
# valid
# copy_from('images_by_category','A', 'pilot_valid', 'A', 20)
# copy_from('images_by_category','C', 'pilot_valid', 'C', 20)
# copy_from('images_by_category','D', 'pilot_valid', 'D', 20)
# copy_from('images_by_category','G', 'pilot_valid', 'G', 20)
# copy_from('images_by_category','H', 'pilot_valid', 'H', 20)
# copy_from('images_by_category','M', 'pilot_valid', 'M', 20)
# copy_from('images_by_category','N', 'pilot_valid', 'N', 20)
# copy_from('images_by_category','O', 'pilot_valid', 'O', 20)

# get_more_images('train_final', 'A', 'train_final', 'A', 'A')
# get_more_images('train_final', 'C', 'train_final', 'C', 'C')
# get_more_images('train_final', 'D', 'train_final', 'D', 'D')
# get_more_images('train_final', 'G', 'train_final', 'G', 'G')
# get_more_images('train_final', 'H', 'train_final', 'H', 'H')
```

```
# get_more_images('train_final', 'M', 'train_final', 'M', 'M')
# get_more_images('train_final', 'N', 'train_final', 'N', 'N')
# get_more_images('train_final', 'O', 'train_final', 'O', 'O')
```

[223]:
```
# train and valid batches for NASNet and VGG16 models

train_batch_nas = ImageDataGenerator(preprocessing_function = tf.keras.
 ↪applications.nasnet.preprocess_input) \
                    .flow_from_directory(directory = '/Users/armaanvalvi/
 ↪Documents/project-data/archive/pilot_folder/',  # directory
                                         target_size = (224,224), # height and width
                                         classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                         batch_size = 10)

valid_batch_nas = ImageDataGenerator(preprocessing_function = tf.keras.
 ↪applications.nasnet.preprocess_input) \
                    .flow_from_directory(directory = '/Users/armaanvalvi/
 ↪Documents/project-data/archive/pilot_valid/',  # directory
                                         target_size = (224,224), # height and width
                                         classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                         batch_size = 5)

train_batch_vgg = ImageDataGenerator(preprocessing_function = tf.keras.
 ↪applications.vgg16.preprocess_input) \
                    .flow_from_directory(directory = '/Users/armaanvalvi/
 ↪Documents/project-data/archive/pilot_folder/',  # directory
                                         target_size = (224,224), # height and width
                                         classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                         batch_size = 10)

valid_batch_vgg = ImageDataGenerator(preprocessing_function = tf.keras.
 ↪applications.vgg16.preprocess_input) \
                    .flow_from_directory(directory = '/Users/armaanvalvi/
 ↪Documents/project-data/archive/pilot_valid/',  # directory
                                         target_size = (224,224), # height and width
                                         classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                         batch_size = 5)
```

```
Found 640 images belonging to 8 classes.
Found 160 images belonging to 8 classes.
Found 640 images belonging to 8 classes.
Found 160 images belonging to 8 classes.
```

0.1.2 Creating train, valid and test sets

[4]:
```
# previously the datasets used

# if os.path.isdir('train_final/A') is False:
#     os.makedirs('train_final/A')
#     os.makedirs('train_final/C')
#     os.makedirs('train_final/D')
#     os.makedirs('train_final/G')
#     os.makedirs('train_final/H')
#     os.makedirs('train_final/M')
#     os.makedirs('train_final/N')
#     os.makedirs('train_final/O')

# if os.path.isdir('valid_final/A') is False:
#     os.makedirs('valid_final/A')
#     os.makedirs('valid_final/C')
#     os.makedirs('valid_final/D')
#     os.makedirs('valid_final/G')
#     os.makedirs('valid_final/H')
#     os.makedirs('valid_final/M')
#     os.makedirs('valid_final/N')
#     os.makedirs('valid_final/O')

# if os.path.isdir('test_final/A') is False:
#     os.makedirs('test_final/A')
#     os.makedirs('test_final/C')
#     os.makedirs('test_final/D')
#     os.makedirs('test_final/G')
#     os.makedirs('test_final/H')
#     os.makedirs('test_final/M')
#     os.makedirs('test_final/N')
#     os.makedirs('test_final/O')
```

[5]:
```
# OLD METHOD

# move 100 images of each type into train_final

# move_from('images_by_category','A', 'train_final', 'A', 100)
# move_from('images_by_category','C', 'train_final', 'C', 100)
# move_from('images_by_category','D', 'train_final', 'D', 100)
# move_from('images_by_category','G', 'train_final', 'G', 100)
# move_from('images_by_category','H', 'train_final', 'H', 100)
# move_from('images_by_category','M', 'train_final', 'M', 100)
# move_from('images_by_category','N', 'train_final', 'N', 100)
# move_from('images_by_category','O', 'train_final', 'O', 100)
```

```
# get 6x data augmentation

# get_more_images('train_final', 'A', 'train_final', 'A', 'A')
# get_more_images('train_final', 'C', 'train_final', 'C', 'C')
# get_more_images('train_final', 'D', 'train_final', 'D', 'D')
# get_more_images('train_final', 'G', 'train_final', 'G', 'G')
# get_more_images('train_final', 'H', 'train_final', 'H', 'H')
# get_more_images('train_final', 'M', 'train_final', 'M', 'M')
# get_more_images('train_final', 'N', 'train_final', 'N', 'N')
# get_more_images('train_final', 'O', 'train_final', 'O', 'O')
```

[6]:
```
# OLD METHOD

# move 20 images of each type into valid_final

# move_from('images_by_category','A', 'valid_final', 'A', 20)
# move_from('images_by_category','C', 'valid_final', 'C', 20)
# move_from('images_by_category','D', 'valid_final', 'D', 20)
# move_from('images_by_category','G', 'valid_final', 'G', 20)
# move_from('images_by_category','H', 'valid_final', 'H', 20)
# move_from('images_by_category','M', 'valid_final', 'M', 20)
# move_from('images_by_category','N', 'valid_final', 'N', 20)
# move_from('images_by_category','O', 'valid_final', 'O', 20)

# get 6x data augmentation

# get_more_images('valid_final', 'A', 'valid_final', 'A', 'A')
# get_more_images('valid_final', 'C', 'valid_final', 'C', 'C')
# get_more_images('valid_final', 'D', 'valid_final', 'D', 'D')
# get_more_images('valid_final', 'G', 'valid_final', 'G', 'G')
# get_more_images('valid_final', 'H', 'valid_final', 'H', 'H')
# get_more_images('valid_final', 'M', 'valid_final', 'M', 'M')
# get_more_images('valid_final', 'N', 'valid_final', 'N', 'N')
# get_more_images('valid_final', 'O', 'valid_final', 'O', 'O')
```

[7]:
```
# OLD METHOD

# move 8 images of each type into test_final

# move_from('images_by_category','A', 'test_final', 'A', 8)
# move_from('images_by_category','C', 'test_final', 'C', 8)
# move_from('images_by_category','D', 'test_final', 'D', 8)
# move_from('images_by_category','G', 'test_final', 'G', 8)
# move_from('images_by_category','H', 'test_final', 'H', 8)
# move_from('images_by_category','M', 'test_final', 'M', 8)
# move_from('images_by_category','N', 'test_final', 'N', 8)
# move_from('images_by_category','O', 'test_final', 'O', 8)
```

```
# get 6x data augmentation

# get_more_images('test_final', 'A', 'test_final', 'A', 'A')
# get_more_images('test_final', 'C', 'test_final', 'C', 'C')
# get_more_images('test_final', 'D', 'test_final', 'D', 'D')
# get_more_images('test_final', 'G', 'test_final', 'G', 'G')
# get_more_images('test_final', 'H', 'test_final', 'H', 'H')
# get_more_images('test_final', 'M', 'test_final', 'M', 'M')
# get_more_images('test_final', 'N', 'test_final', 'N', 'N')
# get_more_images('test_final', 'O', 'test_final', 'O', 'O')
```

[9]:
```
# creating the initial datasets

train_final = ImageDataGenerator(preprocessing_function = tf.keras.applications.
 ↪vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
 ↪project-data/archive/train_final/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                     batch_size = 35)

valid_final = ImageDataGenerator(preprocessing_function = tf.keras.applications.
 ↪vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
 ↪project-data/archive/valid_final/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                     batch_size = 20)

test_final = ImageDataGenerator(preprocessing_function = tf.keras.applications.
 ↪vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
 ↪project-data/archive/test_final/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
 ↪'N', 'O'], # classes for the labels
                                     batch_size = 8)
```

```
Found 5441 images belonging to 8 classes.
Found 1112 images belonging to 8 classes.
Found 447 images belonging to 8 classes.
```

0.1.3 Creating updated training, valid and test data

[10]:
```
if os.path.isdir('train_ultimate/A') is False:
    os.makedirs('train_ultimate/A')
    os.makedirs('train_ultimate/C')
    os.makedirs('train_ultimate/D')
    os.makedirs('train_ultimate/G')
    os.makedirs('train_ultimate/H')
    os.makedirs('train_ultimate/M')
    os.makedirs('train_ultimate/N')
    os.makedirs('train_ultimate/O')

if os.path.isdir('valid_ultimate/A') is False:
    os.makedirs('valid_ultimate/A')
    os.makedirs('valid_ultimate/C')
    os.makedirs('valid_ultimate/D')
    os.makedirs('valid_ultimate/G')
    os.makedirs('valid_ultimate/H')
    os.makedirs('valid_ultimate/M')
    os.makedirs('valid_ultimate/N')
    os.makedirs('valid_ultimate/O')

if os.path.isdir('test_ultimate/A') is False:
    os.makedirs('test_ultimate/A')
    os.makedirs('test_ultimate/C')
    os.makedirs('test_ultimate/D')
    os.makedirs('test_ultimate/G')
    os.makedirs('test_ultimate/H')
    os.makedirs('test_ultimate/M')
    os.makedirs('test_ultimate/N')
    os.makedirs('test_ultimate/O')
```

[ ]:
```
# each folder had at least 2367 augmented images (H_aug had exactly 2367)

# move_to_train('A', 'A', 2367)
# move_to_train('C', 'C', 2367)
# move_to_train('D', 'D', 2367)
# move_to_train('G', 'G', 2367)
# move_to_train('H', 'H', 2367)
# move_to_train('M', 'M', 2367)
# move_to_train('N', 'N', 2367)
# move_to_train('O', 'O', 2367)

# Realized that using 2367 images for each of the 8 classes was causing my␣
 ↪laptop to crash

# copy_from('image_repository','A', 'train_ultimate', 'A', 500)
```

```
# copy_from('image_repository','C', 'train_ultimate', 'C', 500)
# copy_from('image_repository','D', 'train_ultimate', 'D', 500)
# copy_from('image_repository','G', 'train_ultimate', 'G', 500)
# copy_from('image_repository','H', 'train_ultimate', 'H', 500)
# copy_from('image_repository','M', 'train_ultimate', 'M', 500)
# copy_from('image_repository','N', 'train_ultimate', 'N', 500)
# copy_from('image_repository','O', 'train_ultimate', 'O', 500)

# added another 350 images

# move_from('repository','A', 'valid_ultimate', 'A', 200)
# move_from('repository','C', 'valid_ultimate', 'C', 200)
# move_from('repository','D', 'valid_ultimate', 'D', 200)
# move_from('repository','G', 'valid_ultimate', 'G', 200)
# move_from('repository','H', 'valid_ultimate', 'H', 200)
# move_from('repository','M', 'valid_ultimate', 'M', 200)
# move_from('repository','N', 'valid_ultimate', 'N', 200)
# move_from('repository','O', 'valid_ultimate', 'O', 200)

# move_from('repository','A', 'test_ultimate', 'A', 100)
# move_from('repository','C', 'test_ultimate', 'C', 100)
# move_from('repository','D', 'test_ultimate', 'D', 100)
# move_from('repository','G', 'test_ultimate', 'G', 100)
# move_from('repository','H', 'test_ultimate', 'H', 100)
# move_from('repository','M', 'test_ultimate', 'M', 100)
# move_from('repository','N', 'test_ultimate', 'N', 100)
# move_from('repository','O', 'test_ultimate', 'O', 100)
```

[36]:
```
train_ultimate = ImageDataGenerator(preprocessing_function = tf.keras.
↪applications.vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
↪project-data/archive/train_ultimate/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
↪'N', 'O'], # classes for the labels
                                     batch_size = 50)

valid_ultimate = ImageDataGenerator(preprocessing_function = tf.keras.
↪applications.vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
↪project-data/archive/valid_ultimate/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
↪'N', 'O'], # classes for the labels
                                     batch_size = 40)
```

```
test_ultimate = ImageDataGenerator(preprocessing_function = tf.keras.
↪applications.vgg16.preprocess_input) \
                .flow_from_directory(directory = '/Users/armaanvalvi/Documents/
↪project-data/archive/test_ultimate/',  # directory
                                     target_size = (224,224), # height and width
                                     classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
↪'N', 'O'], # classes for the labels
                                     batch_size = 20, shuffle = False)
```

```
Found 6800 images belonging to 8 classes.
Found 1600 images belonging to 8 classes.
Found 800 images belonging to 8 classes.
```

[97]:
```
# train_batches = datagen.flow_from_directory(directory = '/Users/armaanvalvi/
↪Documents/project-data/archive/train_set/',
#                                      target_size = (512,512),
#                                      classes = ['A', 'C', 'D', 'G', 'H', 'M',␣
↪'N', 'O'],
#                                      batch_size = 9)
```

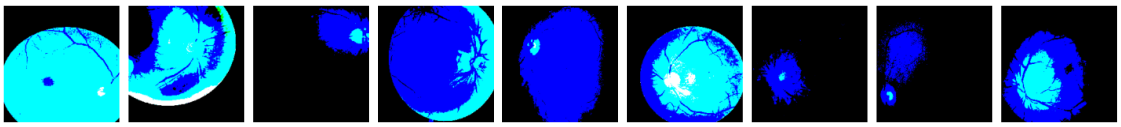
[68]:
```
train_imgs, train_labels = next(train_final) # should be 9 images with the 9␣
↪corresponding labels
```

[69]:
```
# function to plot images in the form of a grid where images are placed
# from the tensorflow website

class_names = ['AMD', 'Cataract', 'Diabetes', 'Glaucoma', 'Hypertension',
               'Myopia', 'Normal', 'Other']

def plotImages(images_arr):
    fig, axes = plt.subplots(1,9, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip(images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
#         plt.xlabel(class_names[train_labels])
    plt.tight_layout()
    plt.show()
```

[70]:
```
plotImages(train_imgs)
print(train_labels)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
```

```
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```



```
[[0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
```

```
[0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.]
[0. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 1.]
[0. 0. 1. 0. 0. 0. 0. 0.]
[0. 1. 0. 0. 0. 0. 0. 0.]]
```

0.1.4 Building and Training a CNN

[71]:
```
# adapted model (with explanations for the parameters) from Deeplizard - thank␣
↪you!

model = Sequential([
    # the first hidden layer - Densely/fully connected layer
    Conv2D(filters = 24,
           kernel_size = (8,8),
           input_shape = (224,224,3), # creating an implicit input layer for␣
↪the model (height,width, no. of colors - RGB)
           activation = 'relu',
           kernel_initializer = 'glorot_uniform', # default value
           padding = 'same'), # zero padding to ensure the dimensionality of␣
↪the images isn't reduced after the convolution operations)
                              # otherwise, the default is padding = 'valid',␣
↪which does not pad the input for each indiv layer
    MaxPool2D(pool_size = (4,4), strides = 4, padding = 'valid'), # cut pooling␣
↪dimensions by half to shrink the number of trainable parameters
    Conv2D(filters = 48, # following common practice of increasing functions as␣
↪you have more hidden layers
           kernel_size = (8,8),
           activation = 'relu',
           padding = 'same',
           use_bias = True, # default = True
           bias_initializer = 'zeros'), # default = 'zeros'
    MaxPool2D(pool_size = (4,4), strides = 4),
    Flatten(), # before passing output from a convo layer to a dense layer, you␣
↪have to turn the output into a one-dim tensor by flattening the output by␣
↪multiplying the dimensions of the data from the conv layer by the filters in␣
↪that layer
    Dropout(0.3), # deactivate 30% of the neurons
    Dense(units = 8, activation = 'sigmoid')]) # the output layer
```

[132]:
```
# model.summary()
```

[72]:
```
# now that model is built, need to prepare it for training with model.compile

# model.compile(optimizer=SGD(learning_rate = 0.001, momentum = 0.8, decay = 0.
 ↪001/10, nesterov = False
#                              ),
#               loss = 'categorical_crossentropy',
#               metrics = ['accuracy'
#                         ])

model.compile(optimizer=Adadelta(learning_rate = 0.1
                             ),
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'
                        ])
```

[73]:
```
# train the model with model.fit
model.fit(x = train_final, validation_data = valid_final, epochs = 10, verbose␣
 ↪= 2)
```

```
Train for 156 steps, validate for 56 steps
Epoch 1/10
156/156 - 251s - loss: 2.2053 - accuracy: 0.1230 - val_loss: 2.0794 -
val_accuracy: 0.1250
Epoch 2/10
156/156 - 228s - loss: 2.0930 - accuracy: 0.1239 - val_loss: 2.0794 -
val_accuracy: 0.1250
Epoch 3/10
156/156 - 243s - loss: 2.0883 - accuracy: 0.1230 - val_loss: 2.0794 -
val_accuracy: 0.1250
Epoch 4/10
156/156 - 252s - loss: 2.1527 - accuracy: 0.1224 - val_loss: 2.0861 -
val_accuracy: 0.1250
Epoch 5/10
156/156 - 229s - loss: 2.0885 - accuracy: 0.1226 - val_loss: 2.0794 -
val_accuracy: 0.1250
Epoch 6/10
156/156 - 249s - loss: 2.0841 - accuracy: 0.1233 - val_loss: 2.1269 -
val_accuracy: 0.1250
Epoch 7/10
156/156 - 252s - loss: 2.1263 - accuracy: 0.1253 - val_loss: 2.0858 -
val_accuracy: 0.1250
Epoch 8/10
156/156 - 257s - loss: 2.0951 - accuracy: 0.1226 - val_loss: 2.0794 -
val_accuracy: 0.1250
Epoch 9/10
156/156 - 265s - loss: 2.0902 - accuracy: 0.1244 - val_loss: 2.0794 -
val_accuracy: 0.1250
```

```
Epoch 10/10
156/156 - 252s - loss: 2.1010 - accuracy: 0.1246 - val_loss: 2.0799 -
val_accuracy: 0.1223
```

[73]:
```
<tensorflow.python.keras.callbacks.History at 0x7fc197b76590>
```

[ ]:
```
# look into this
# tf.keras.callbacks.EarlyStopping (documentation).
# Alert when a certain condition is met; if the val_loss hasn't decreased in 5␣
 ↪epochs then

# early_stop = EarlyStopping(monitor='val_loss', patience=5)
# fit_generator(... callbacks=[es])
```

[ ]:
```
# let's test the model for inference using the test data
# how we can use CNN for inference using the keras.Sequential API

# test_imgs, test_labels = next(test_final) #'next' returns the next item in␣
 ↪the iterator
# plotImages(test_imgs) # which we defined
# print(test_labels)
```

0.1.5 Build and Train Fine-Tuned Trained Models

VGG16 Model

[28]:
```
vgg16_model = tf.keras.applications.vgg16.VGG16()
# vgg16_model.summary()
```

[29]:
```
type(dir(vgg16_model))
```

[29]:
```
list
```

[30]:
```
# to look at all the attributes/methods of the object pertaining to layers
pd.Series(dir(vgg16_model))[pd.Series(dir(vgg16_model)).str.contains('layer',␣
 ↪regex = True)]
```

[30]:
```
100          _input_layers
101         _insert_layers
104              _is_layer
107    _layer_call_argspecs
108                 _layers
140         _output_layers
187          _track_layers
231              get_layer
245                 layers
dtype: object
```

[31]:
```
vgg16_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
```

```
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

[44]:
```
def prepare_model(model, activation_function, dropout_rate):
    new_model = Sequential()

    # loop through every layer except for the last (since the output and number␣
 ↪of classes are different)
    # and add each layer to the Sequential model
    for layer in model.layers[:-1]:
        new_model.add(layer)
        if layer.name in ['fc1', 'fc2']:
            new_model.add(Dropout(dropout_rate))

    # iterate over all the layers in the Sequential model
    for layer in new_model.layers:
        layer.trainable = False # freeze the trainable parameters (weights and␣
 ↪biases)

    new_model.add(Dense(units=8, activation = activation_function))

    return new_model
```

[45]:
```
final_model = prepare_model(vgg16_model, 'softmax', dropout_rate = 0.2)
```

NASNetMobile Model

[13]:
```
nasNet_model = tf.keras.applications.nasnet.NASNetMobile(
    input_shape=(224,224,3), include_top=False, weights='imagenet', pooling =␣
 ↪max, classes = 8)
```

[14]:
```
# thank you EngrStudent and Dr.Snoopy on StackFlow!
```

[96]:
```
# Dense model
my_inputs = Input(shape = (224,224,3))
hidden_1 = Dense(units = 8, activation = 'relu')(my_inputs)

# creating output layer
hidden_2 = Dense(units = np.product((224,224,3)), activation =␣
 ↪'sigmoid')(hidden_1)
```

```
transformed = keras.layers.Reshape((224,224,3),)(hidden_2)
dense_model = Model(inputs = my_inputs, outputs = transformed)
```

[15]:
```
# hidden_final = Dense(8, activation = 'softmax')(nasNet_model.layers[-2].
 ↪output)
# transformed_final = keras.layers.Reshape((224,224,3),)(hidden_final)
# dense_final = Model(inputs = my_inputs, outputs = transformed_final)
```

[160]:
```
# prepend dense model to nasNet_model

inp = Input(shape = (224,224,3))
x = dense_model(inp)
x = nasNet_model(x)
the_final_model = Sequential(inp, x)
```

[201]:
```
# adding an output layer to the nasNet model to make changes to the final model
# Thanks CVxTz on Kaggle and Program Creek!
# https://www.programcreek.com/python/example/89688/keras.layers.
 ↪GlobalAveragePooling2D
y = the_final_model(inp)
out1 = GlobalMaxPooling2D()(y)
out2 = GlobalAveragePooling2D()(y)
out3 = Flatten()(y)
out = Concatenate(axis = -1)([out1, out2, out3])
out = Dropout(0.5)(out)
out = Dense(8, activation = "softmax")(out)
the_model = Model(inp, out)
the_model.compile(optimizer=Adam(0.1), loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

0.1.6 Train the fine-tuned models

[47]:
```
# compile the model
final_model.compile(optimizer = Adadelta(learning_rate = 0.1),
                                    loss = 'categorical_crossentropy',
                                    metrics = ['accuracy'])
```

[48]:
```
# fit the model as always to train the model (passing in the train and valid␣
 ↪data)
final_model.fit(x = train_ultimate,
          validation_data = valid_ultimate,
          epochs = 30, verbose = 2)
```

```
Train for 136 steps, validate for 40 steps
Epoch 1/30
136/136 - 2641s - loss: 2.6980 - accuracy: 0.1299 - val_loss: 2.2419 -
val_accuracy: 0.1431
```

```
Epoch 2/30
136/136 - 2705s - loss: 2.5271 - accuracy: 0.1422 - val_loss: 2.2048 -
val_accuracy: 0.1456
Epoch 3/30
136/136 - 2618s - loss: 2.4819 - accuracy: 0.1334 - val_loss: 2.1548 -
val_accuracy: 0.1506
Epoch 4/30
136/136 - 18763s - loss: 2.3959 - accuracy: 0.1532 - val_loss: 2.1477 -
val_accuracy: 0.1619
Epoch 5/30
136/136 - 2455s - loss: 2.3945 - accuracy: 0.1571 - val_loss: 2.1222 -
val_accuracy: 0.1656
Epoch 6/30
136/136 - 2464s - loss: 2.3313 - accuracy: 0.1607 - val_loss: 2.1427 -
val_accuracy: 0.1612
Epoch 7/30
136/136 - 2714s - loss: 2.2952 - accuracy: 0.1815 - val_loss: 2.1157 -
val_accuracy: 0.1606
Epoch 8/30
136/136 - 5511s - loss: 2.2662 - accuracy: 0.1679 - val_loss: 2.1007 -
val_accuracy: 0.1800
Epoch 9/30
136/136 - 2729s - loss: 2.2449 - accuracy: 0.1696 - val_loss: 2.1018 -
val_accuracy: 0.1719
Epoch 10/30
136/136 - 3665s - loss: 2.2293 - accuracy: 0.1826 - val_loss: 2.0937 -
val_accuracy: 0.1731
Epoch 11/30
136/136 - 3275s - loss: 2.1949 - accuracy: 0.1887 - val_loss: 2.0968 -
val_accuracy: 0.1612
Epoch 12/30
136/136 - 2876s - loss: 2.1663 - accuracy: 0.1937 - val_loss: 2.0824 -
val_accuracy: 0.1931
Epoch 13/30
136/136 - 2961s - loss: 2.1615 - accuracy: 0.1940 - val_loss: 2.0845 -
val_accuracy: 0.1838
Epoch 14/30
136/136 - 3084s - loss: 2.1300 - accuracy: 0.2071 - val_loss: 2.0855 -
val_accuracy: 0.1831
Epoch 15/30
136/136 - 3210s - loss: 2.1101 - accuracy: 0.2206 - val_loss: 2.0662 -
val_accuracy: 0.1937
Epoch 16/30
136/136 - 2840s - loss: 2.1005 - accuracy: 0.2197 - val_loss: 2.0575 -
val_accuracy: 0.1912
Epoch 17/30
136/136 - 2606s - loss: 2.0815 - accuracy: 0.2231 - val_loss: 2.0699 -
val_accuracy: 0.1750
```

```
Epoch 18/30
136/136 - 2535s - loss: 2.0685 - accuracy: 0.2221 - val_loss: 2.0754 -
val_accuracy: 0.1969
Epoch 19/30
136/136 - 2579s - loss: 2.0477 - accuracy: 0.2275 - val_loss: 2.0624 -
val_accuracy: 0.1912
Epoch 20/30
136/136 - 2528s - loss: 2.0242 - accuracy: 0.2331 - val_loss: 2.0692 -
val_accuracy: 0.1844
Epoch 21/30
136/136 - 2617s - loss: 2.0130 - accuracy: 0.2415 - val_loss: 2.0635 -
val_accuracy: 0.1856
Epoch 22/30
136/136 - 2695s - loss: 1.9981 - accuracy: 0.2443 - val_loss: 2.0646 -
val_accuracy: 0.2013
Epoch 23/30
136/136 - 2614s - loss: 1.9929 - accuracy: 0.2504 - val_loss: 2.0569 -
val_accuracy: 0.1988
Epoch 24/30
136/136 - 2703s - loss: 1.9717 - accuracy: 0.2482 - val_loss: 2.0486 -
val_accuracy: 0.1925
Epoch 25/30
136/136 - 2740s - loss: 1.9620 - accuracy: 0.2512 - val_loss: 2.0653 -
val_accuracy: 0.1950
Epoch 26/30
136/136 - 2649s - loss: 1.9565 - accuracy: 0.2621 - val_loss: 2.0420 -
val_accuracy: 0.2062
Epoch 27/30
136/136 - 2620s - loss: 1.9310 - accuracy: 0.2696 - val_loss: 2.0531 -
val_accuracy: 0.2006
Epoch 28/30
136/136 - 2621s - loss: 1.9272 - accuracy: 0.2626 - val_loss: 2.0602 -
val_accuracy: 0.1969
Epoch 29/30
136/136 - 20126s - loss: 1.9183 - accuracy: 0.2744 - val_loss: 2.0499 -
val_accuracy: 0.1956
Epoch 30/30
136/136 - 2622s - loss: 1.9084 - accuracy: 0.2782 - val_loss: 2.0354 -
val_accuracy: 0.2019
```

[48]:
```
<tensorflow.python.keras.callbacks.History at 0x7ff206d75d90>
```

[49]:
```
# final_model.save("VGG16_ocular.h5")
```

[51]:
```
# Do Inference by passing in the test data
predictions = final_model.predict(x = test_ultimate, verbose = 0)
```

[52]:
```
cm = confusion_matrix(y_true = test_ultimate.classes,
                      y_pred = np.argmax(predictions, axis = -1))
```

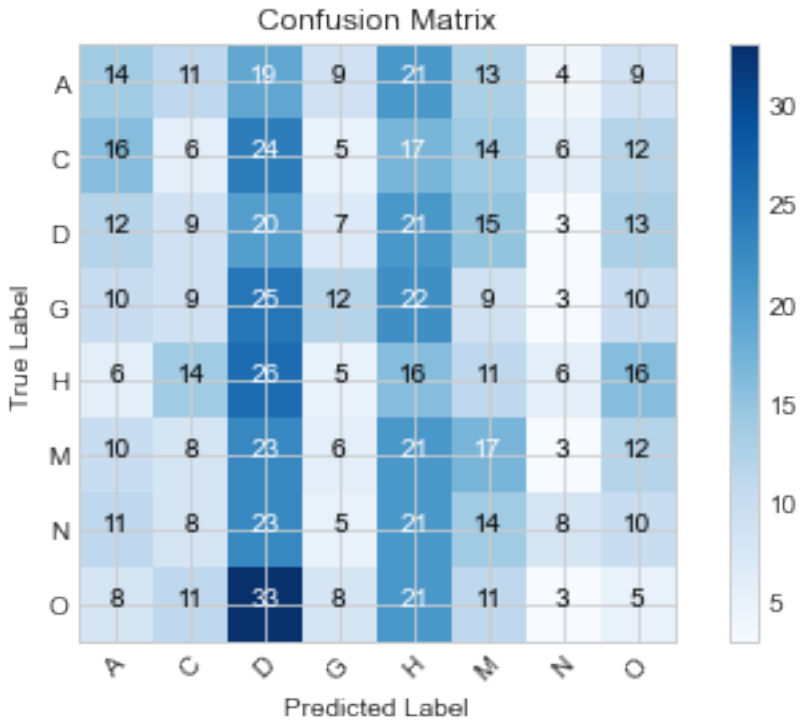
[53]:
```
test_ultimate.class_indices # to see the order of our class indices
```

[53]:
```
{'A': 0, 'C': 1, 'D': 2, 'G': 3, 'H': 4, 'M': 5, 'N': 6, 'O': 7}
```

[55]:
```
# from the sklearn website

def plot_confusion_matrix(cm, classes,
                          normalize = False,
                          title = 'Confusion Matrix',
                          cmap = plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting 'normalize = True'
    """
    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max()/2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j,i,cm[i,j],
                 horizontalalignment = 'center',
                 color = 'white' if cm[i,j] > thresh else 'black')

        plt.tight_layout()
        plt.ylabel('True Label')
        plt.xlabel('Predicted Label')
```

[56]:
```
cm_plot_labels = ['A','C', 'D', 'G', 'H', 'M', 'N', 'O']
plot_confusion_matrix(cm = cm,
                      classes = cm_plot_labels,
                      title = 'Confusion Matrix')
```

```
Confusion matrix, without normalization
[[14 11 19  9 21 13  4  9]
 [16  6 24  5 17 14  6 12]
 [12  9 20  7 21 15  3 13]
 [10  9 25 12 22  9  3 10]
 [ 6 14 26  5 16 11  6 16]
 [10  8 23  6 21 17  3 12]
 [11  8 23  5 21 14  8 10]
 [ 8 11 33  8 21 11  3  5]]
```



0.1.7 Possible Improvements

[ ]:
```
# although 8000 images is quite substantial when not using a GPU, there is␣
 ↪always scope for improving the outcome
# by increasing the number of unique images
```

[ ]:
```
# treating each augmented image as another retinal scan from a patient
# it might be that the trained data has many variants of the same retinal scan
# this can potentially skew the results
```

[ ]:
```

```