OcularCNN-Improvements

In [1]:
```python
import pandas as pd
import random
import numpy as np

theFile = pd.read_csv("../data/full_df.csv")
theFile
```

Out[1]:

	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C
0	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fundus	0	0	0	1
1	1	57	Male	1_left.jpg	1_right.jpg	normal fundus	normal fundus	1	0	0	0
2	2	42	Male	2_left.jpg	2_right.jpg	laser spot, moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0
3	4	53	Male	4_left.jpg	4_right.jpg	macular epiretinal membrane	mild nonproliferative retinopathy	0	1	0	0
4	5	50	Female	5_left.jpg	5_right.jpg	moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...
6387	4686	63	Male	4686_left.jpg	4686_right.jpg	severe nonproliferative retinopathy	proliferative diabetic retinopathy	0	1	0	0
6388	4688	42	Male	4688_left.jpg	4688_right.jpg	moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0
6389	4689	54	Male	4689_left.jpg	4689_right.jpg	mild nonproliferative retinopathy	normal fundus	0	1	0	0
6390	4690	57	Male	4690_left.jpg	4690_right.jpg	mild nonproliferative retinopathy	mild nonproliferative retinopathy	0	1	0	0
6391	4784	58	Male	4784_left.jpg	4784_right.jpg	hypertensive retinopathy, age-related macular d...	hypertensive retinopathy, age-related macular d...	0	0	0	0

6392 rows × 19 columns

In [2]:
```python
theFile.drop(np.linspace(1, 3194, 3193, dtype = int), inplace = True )
target = theFile.loc[(theFile['Left-Diagnostic Keywords'] == 'cataract') | (theFile['L


target = theFile.loc[(theFile['Left-Diagnostic Keywords'] == 'cataract') | (theFile['L
```

In [3]:
```python
print(len(target.loc[target['Left-Diagnostic Keywords'] == 'cataract']))
print(len(target.loc[target['Left-Diagnostic Keywords'] == 'normal fundus']))
```

```
135
1433
```

In [4]:
```python
#As shown above, there are much more normal fundus entries than cataract
#Let's sample the normal fundus to get equivalent numbers of each class
cataract = target.loc[target['Left-Diagnostic Keywords'] == 'cataract']
normal = target.loc[target['Left-Diagnostic Keywords'] == 'normal fundus']
normal = normal.sample(135)
frames = [cataract, normal]
target = pd.concat(frames)

#now shuffle these randomly
# shuffle the DataFrame rows
target= target.sample(frac = 1)
```

In [5]:
```python
pictures = target[['filename']]
```

In [6]:
```python
target
```

Out[6]:

	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A
4058	990	41	Female	990_left.jpg	990_right.jpg	normal fundus	macular epiretinal membrane	0	0	0	0	0
6232	4496	53	Female	4496_left.jpg	4496_right.jpg	normal fundus	mild nonproliferative retinopathy	0	1	0	0	0
4132	1113	56	Female	1113_left.jpg	1113_right.jpg	normal fundus	mild nonproliferative retinopathy	0	1	0	0	0
4803	2432	45	Female	2432_left.jpg	2432_right.jpg	normal fundus	normal fundus	1	0	0	0	0
4588	2123	47	Male	2123_left.jpg	2123_right.jpg	cataract	refractive media opacity	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
5080	2740	55	Male	2740_left.jpg	2740_right.jpg	normal fundus	normal fundus	1	0	0	0	0
4574	2109	82	Male	2109_left.jpg	2109_right.jpg	cataract	cataract	0	0	0	1	0
4585	2120	72	Female	2120_left.jpg	2120_right.jpg	cataract	cataract	0	0	0	1	0
5243	2916	61	Male	2916_left.jpg	2916_right.jpg	normal fundus	normal fundus	1	0	0	0	0
4634	2172	73	Male	2172_left.jpg	2172_right.jpg	cataract	normal fundus	0	0	0	1	0

270 rows × 19 columns

In [7]:
```python
classnames = ['[\'N\']', '[\'D\']','[\'G\']','[\'C\']','[\'A\']','[\'H\']','[\'M\']',
classnames
```

Out[7]:
```
["['N']", "['D']", "['G']", "['C']", "['A']", "['H']", "['M']", "['O']"]
```

In [8]:
```python
pictures = pictures.values.tolist()
pictures = sum(pictures, [])
pictures
```

Out[8]:
```
['990_left.jpg',
 '4496_left.jpg',
 '1113_left.jpg',
 '2432_left.jpg',
 '2123_left.jpg',
 '2204_left.jpg',
 '3120_left.jpg',
 '1038_left.jpg',
 '149_left.jpg',
 '2225_left.jpg',
 '2708_left.jpg',
 '448_left.jpg',
 '683_left.jpg',
 '2236_left.jpg',
 '3204_left.jpg',
 '2532_left.jpg',
 '384_left.jpg',
 '3394_left.jpg',
 '2209_left.jpg',
 '2832_left.jpg',
 '2135_left.jpg',
 '2507_left.jpg',
 '4219_left.jpg',
 '2855_left.jpg',
 '2932_left.jpg',
 '2101_left.jpg',
 '3153_left.jpg',
 '2262_left.jpg',
 '2145_left.jpg',
 '2216_left.jpg',
 '2846_left.jpg',
 '2171_left.jpg',
 '2157_left.jpg',
 '2140_left.jpg',
 '3420_left.jpg',
 '3163_left.jpg',
 '85_left.jpg',
 '3183_left.jpg',
 '2117_left.jpg',
 '2744_left.jpg',
 '2134_left.jpg',
 '2166_left.jpg',
 '2098_left.jpg',
 '2774_left.jpg',
 '1484_left.jpg',
 '2232_left.jpg',
 '907_left.jpg',
 '2543_left.jpg',
 '4531_left.jpg',
 '3311_left.jpg',
 '2470_left.jpg',
 '2185_left.jpg',
 '2153_left.jpg',
 '69_left.jpg',
 '2211_left.jpg',
 '2194_left.jpg',
 '2130_left.jpg',
 '2189_left.jpg',
 '2129_left.jpg',
 '2243_left.jpg',
 '3208_left.jpg',
 '2666_left.jpg',
 '999_left.jpg',
 '3172_left.jpg',
 '2452_left.jpg',
 '2872_left.jpg',
 '4209_left.jpg',
 '294_left.jpg',
 '2181_left.jpg',
 '3361_left.jpg',
 '2649_left.jpg',
 '3076_left.jpg',
 '2124_left.jpg',
 '3064_left.jpg',
 '2127_left.jpg',
 '2151_left.jpg',
 '103_left.jpg',
 '2095_left.jpg',
 '2461_left.jpg',
 '0_right.jpg',
 '2178_left.jpg',
 '2179_left.jpg',
 '809_left.jpg',
 '2692_left.jpg',
 '824_left.jpg',
 '2200_left.jpg',
 '1162_left.jpg',
 '477_left.jpg',
 '702_left.jpg',
 '2466_left.jpg',
 '2227_left.jpg',
 '2861_left.jpg',
 '1452_left.jpg',
 '2183_left.jpg',
 '160_left.jpg',
 '2158_left.jpg',
 '2180_left.jpg',
 '2207_left.jpg',
 '1415_left.jpg',
 '333_left.jpg',
 '1166_left.jpg',
 '2152_left.jpg',
 '2529_left.jpg',
 '3379_left.jpg',
 '2184_left.jpg',
 '877_left.jpg',
 '2849_left.jpg',
 '2176_left.jpg',
 '599_left.jpg',
 '2575_left.jpg',
 '2965_left.jpg',
 '2233_left.jpg',
 '2221_left.jpg',
 '1164_left.jpg',
 '2819_left.jpg',
 '2369_left.jpg',
 '691_left.jpg',
 '2150_left.jpg',
 '750_left.jpg',
 '695_left.jpg',
 '3057_left.jpg',
 '3025_left.jpg',
 '3112_left.jpg',
 '1444_left.jpg',
 '2126_left.jpg',
 '2146_left.jpg',
 '2247_left.jpg',
 '3423_left.jpg',
 '2136_left.jpg',
 '2377_left.jpg',
 '2195_left.jpg',
 '2099_left.jpg',
 '2346_left.jpg',
 '629_left.jpg',
 '2154_left.jpg',
 '2167_left.jpg',
 '2144_left.jpg',
 '2155_left.jpg',
 '2122_left.jpg',
 '2514_left.jpg',
 '4230_left.jpg',
 '2110_left.jpg',
 '2107_left.jpg',
 '4224_left.jpg',
 '1008_left.jpg',
 '3173_left.jpg',
 '2993_left.jpg',
 '2441_left.jpg',
 '347_left.jpg',
 '820_left.jpg',
 '2170_left.jpg',
 '2105_left.jpg',
 '2234_left.jpg',
 '2223_left.jpg',
 '1468_left.jpg',
 '4500_left.jpg',
 '2113_left.jpg',
 '2212_left.jpg',
 '553_left.jpg',
 '3102_left.jpg',
 '2480_left.jpg',
 '3128_left.jpg',
 '3121_left.jpg',
 '2722_left.jpg',
 '2375_left.jpg',
 '2876_left.jpg',
 '2448_left.jpg',
 '2121_left.jpg',
 '2163_left.jpg',
 '4577_left.jpg',
 '2215_left.jpg',
 '1102_left.jpg',
 '2434_left.jpg',
 '2138_left.jpg',
 '2175_left.jpg',
 '1200_left.jpg',
 '2206_left.jpg',
 '2235_left.jpg',
 '2169_left.jpg',
 '2931_left.jpg',
 '2446_left.jpg',
 '147_left.jpg',
 '2914_left.jpg',
 '2462_left.jpg',
 '2904_left.jpg',
 '2246_left.jpg',
 '2133_left.jpg',
 '3136_left.jpg',
 '3020_left.jpg',
 '3159_left.jpg',
 '945_left.jpg',
 '3067_left.jpg',
 '2182_left.jpg',
 '2385_left.jpg',
 '2116_left.jpg',
 '2128_left.jpg',
 '2210_left.jpg',
 '2890_left.jpg',
 '173_left.jpg',
 '2132_left.jpg',
 '191_left.jpg',
 '2219_left.jpg',
 '2239_left.jpg',
 '4309_left.jpg',
 '2111_left.jpg',
 '3401_left.jpg',
 '1987_left.jpg',
 '2108_left.jpg',
 '2399_left.jpg',
 '2192_left.jpg',
 '16_left.jpg',
 '2137_left.jpg',
 '1115_left.jpg',
 '3331_left.jpg',
 '2217_left.jpg',
 '2198_left.jpg',
 '439_left.jpg',
 '58_left.jpg',
 '1285_left.jpg',
 '2104_left.jpg',
 '4245_left.jpg',
 '2159_left.jpg',
 '2114_left.jpg',
 '2125_left.jpg',
 '2240_left.jpg',
 '2431_left.jpg',
 '3129_left.jpg',
 '611_left.jpg',
 '2482_left.jpg',
 '425_left.jpg',
 '2349_left.jpg',
 '4489_left.jpg',
 '2242_left.jpg',
 '330_left.jpg',
 '2149_left.jpg',
 '1435_left.jpg',
 '2778_left.jpg',
 '3249_left.jpg',
 '2147_left.jpg',
 '1483_left.jpg',
 '2168_left.jpg',
 '187_left.jpg',
 '2142_left.jpg',
 '2902_left.jpg',
 '1144_left.jpg',
 '3374_left.jpg',
 '119_left.jpg',
 '3075_left.jpg',
 '8_left.jpg',
 '2896_left.jpg',
 '560_left.jpg',
 '2609_left.jpg',
 '2156_left.jpg',
 '3382_left.jpg',
 '3424_left.jpg',
 '317_left.jpg',
 '2987_left.jpg',
 '2398_left.jpg',
 '2226_left.jpg',
 '726_left.jpg',
 '2199_left.jpg',
 '254_left.jpg',
 '3303_left.jpg',
 '2203_left.jpg',
 '3448_left.jpg',
 '2740_left.jpg',
 '2109_left.jpg',
 '2120_left.jpg',
 '2916_left.jpg',
 '2172_left.jpg']
```

In [9]:
```python
#sampling 30 photos
small_pics= random.sample(pictures, 270)
```

In [ ]:

In [10]:
```python
#grab target if label matches
classifications=list()
for pic_name in small_pics:
    intermediate = theFile.loc[theFile['filename'] == pic_name]
    label = intermediate['Left-Diagnostic Keywords'].to_string().replace('\'','').repl
    label = ''.join([i for i in label if not i.isdigit()])
    classifications.append(label)
#classifications contains the "target" corresponding to small_pics
```

In [11]:
```python
classifications
```

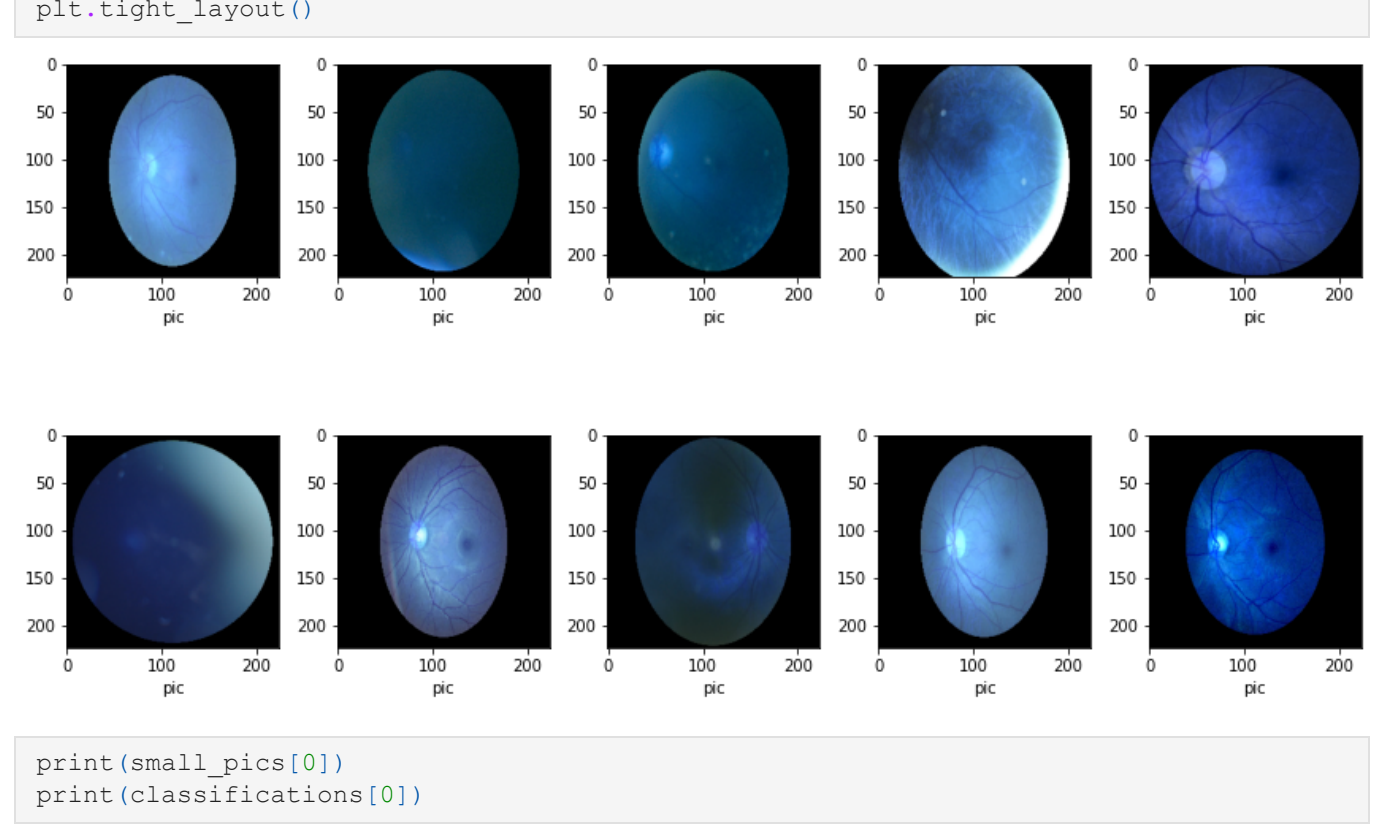
Out[11]:
```
['cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'normal fundus',
 'normal fundus',
 'normal fundus',
 'cataract',
 'cataract',
 'normal fundus',
 'cataract',
 'normal fundus',
 'cataract']
```

In [12]:
```python
import matplotlib.pyplot as plt
import cv2
actual_pics = []

img_res = 224

for file in small_pics:
    img = cv2.imread('../Challenge/images/' + file) #zero makes it greyscale
    #this is resizing all images
    img = cv2.resize(img, (img_res, img_res))
    actual_pics.append(img)
```

In [13]:
```python
actual_pics[0]
```

Out[13]:
```
array([[[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       ...,

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]]], dtype=uint8)
```

In [14]:
```python
plt.figure(figsize=(12,7))
for i in range(10):
    sample = random.choice(range(len(actual_pics)))
    image = actual_pics[sample]

    plt.subplot(2,5,i+1)
    plt.imshow(image)
    plt.xlabel('pic')
plt.tight_layout()
```



In [15]:
```python
print(small_pics[0])
print(classifications[0])
```

```
2204_left.jpg
cataract
```

In [16]:
```python
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras import utils

#normalize our data !!
labels = {
    "cataract": 0,
    "normal fundus": 1
}

labels = np.array(labels)


for i in range(len(classifications)):
        if(classifications[i] == 'cataract'):
                classifications[i] = 0
        elif(classifications[i] == 'normal fundus'):
                 classifications[i] = 1

classifications = utils.to_categorical(classifications)
```

In [17]:
```python
actual_pics = np.array(actual_pics)
classifications = np.array(classifications)

classifications = np.array(classifications)
actual_pics.astype('float32')/255.0

#shuffle = false prevents data split being different everytime
X_train, X_test, y_train, y_test = train_test_split(actual_pics, classifications, test

#split test into validate and test, again making sure the data is always the same for
X_train, xt, y_train, yt = train_test_split(X_train, y_train, test_size=0.1, shuffle =
```

In [18]:
```python
#begin NN
import logging
logging.basicConfig()
import tensorflow as tf
```

In [19]:
```python
import numpy
len(actual_pics[2][0])
```

Out[19]:
```
224
```

In [20]:
```python
print(X_train.shape)
y_train.shape
```

```
(194, 224, 224, 3)
```

Out[20]:
```
(194, 2)
```

In [21]:
```python
# convert to numpy arrays
x = np.array(X_train)
y = np.array(y_train)

X_test = np.array(X_test)
y_test = np.array(y_test)

xt = np.array(xt)
yt = np.array(yt)
```

In [22]:
```python
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dropout
from keras.layers.advanced_activations import LeakyReLU

model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation = 'linear', input_shape=(img_res, img_re
model.add(LeakyReLU(alpha=0.1))
model.add(layers.MaxPooling2D(2,2))
model.add(Dropout(0.25))
model.add(layers.Conv2D(128, (3,3), activation = 'linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(layers.MaxPooling2D(2,2))
model.add(Dropout(0.25))
model.add(layers.Conv2D(128, (3,3), activation = 'linear'))
model.add(LeakyReLU(alpha=0.1))
model.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation = 'sigmoid'))
model.add(layers.Dense(2, activation='softmax'))
model.summary()
from tensorflow import keras
opt = keras.optimizers.Adam( learning_rate=0.00001)

#early stopping stuff
# simple early stopping
from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=20, min_delta=(

model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 222, 222, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0
_________________________________________________________________
dropout (Dropout)            (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 109, 109, 128)     36992
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 109, 109, 128)     0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 54, 54, 128)       0
_________________________________________________________________
dropout_1 (Dropout)          (None, 54, 54, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 52, 52, 128)       147584
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 52, 52, 128)       0
```

```
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 86528)             0
_________________________________________________________________
dense (Dense)                (None, 128)               11075712
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 11,261,442
Trainable params: 11,261,442
Non-trainable params: 0
_________________________________________________________________
```

In [23]:

```
print(np.size(x))
print(np.shape(x))
print(np.size(y))
print(np.shape(y))
```

```
29202432
(194, 224, 224, 3)
388
(194, 2)
```

In [24]:

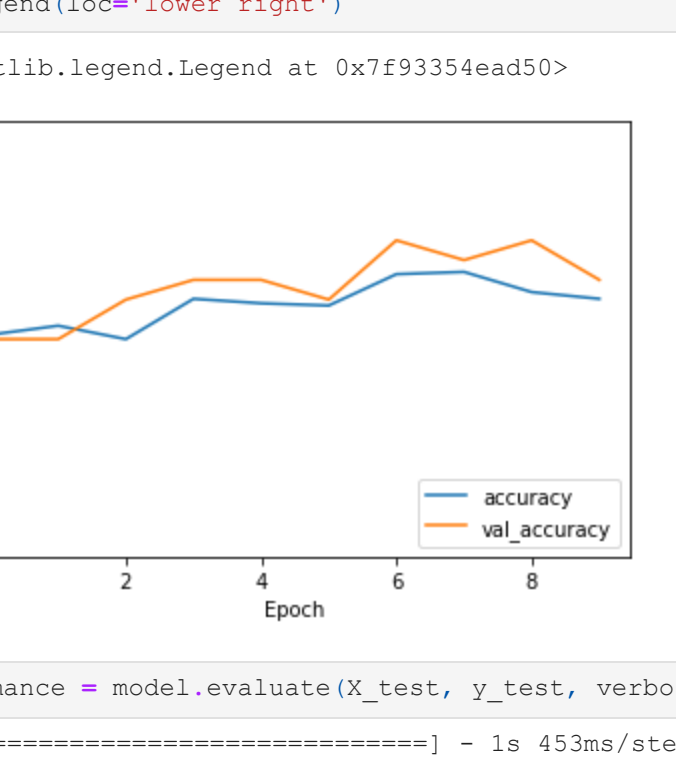
```
#this stores the fitting information
#put in our own data for the images and labelling stuff

history = model.fit(x, y, epochs = 10, validation_data = (xt, yt), callbacks = [es])

#dont forget to keep training until your accuracy becomes bad
```

```
Epoch 1/10
7/7 [==============================] - 19s 3s/step - loss: 0.7603 - accuracy: 0.5463 - val_loss: 0.6945 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 13s 2s/step - loss: 0.7005 - accuracy: 0.5379 - val_loss: 0.6847 - val_accuracy: 0.5000
Epoch 3/10
7/7 [==============================] - 14s 2s/step - loss: 0.6844 - accuracy: 0.5163 - val_loss: 0.6784 - val_accuracy: 0.5909
Epoch 4/10
7/7 [==============================] - 14s 2s/step - loss: 0.6737 - accuracy: 0.6050 - val_loss: 0.6769 - val_accuracy: 0.6364
Epoch 5/10
7/7 [==============================] - 14s 2s/step - loss: 0.6700 - accuracy: 0.6041 - val_loss: 0.6708 - val_accuracy: 0.6364
Epoch 6/10
7/7 [==============================] - 13s 2s/step - loss: 0.6632 - accuracy: 0.6111 - val_loss: 0.6727 - val_accuracy: 0.5909
Epoch 7/10
7/7 [==============================] - 14s 2s/step - loss: 0.6718 - accuracy: 0.6098 - val_loss: 0.6675 - val_accuracy: 0.7273
Epoch 8/10
7/7 [==============================] - 13s 2s/step - loss: 0.6511 - accuracy: 0.6498 - val_loss: 0.6688 - val_accuracy: 0.6818
Epoch 9/10
7/7 [==============================] - 13s 2s/step - loss: 0.6641 - accuracy: 0.5980 - val_loss: 0.6722 - val_accuracy: 0.7273
Epoch 10/10
7/7 [==============================] - 14s 2s/step - loss: 0.6652 - accuracy: 0.5865 - val_loss: 0.6655 - val_accuracy: 0.6364
```

In [25]:

```
# Plot the change in accuracy and validation accruacy as a function of epochs

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.legend(loc='lower right')
```

Out[25]: <matplotlib.legend.Legend at 0x7f93354ead50>



In [26]:

```
performance = model.evaluate(X_test, y_test, verbose=1)
```

```
2/2 [==============================] - 1s 453ms/step - loss: 0.6503 - accuracy: 0.6667
```

In [27]:

```
y_pred = model.predict(X_test)
print(y_pred)
```

```
[[0.5788577  0.42114228]
 [0.97771764 0.02228238]
 [0.95906943 0.0409306 ]
 [0.50533015 0.49466982]
 [0.68056893 0.31943107]
 [0.5151514  0.48484862]
 [0.5313143  0.46868566]
 [0.5827789  0.4172211 ]
 [0.58487195 0.4151281 ]
 [0.5960251  0.4039748 ]
 [0.60610163 0.3938983 ]
 [0.5258783  0.47412172]
 [0.536712   0.46328804]
 [0.5292513  0.4707487 ]
 [0.5387855  0.46121445]
 [0.5465227  0.45347735]
 [0.33974645 0.6602535 ]
 [0.5355231  0.46447688]
 [0.58430666 0.41569328]
 [0.57782865 0.42217132]
 [0.7352031  0.26479688]
 [0.6021158  0.39788422]
 [0.57931334 0.4206867 ]
 [0.49606153 0.50393844]
 [0.58762383 0.4123762 ]
 [0.6887743  0.31122577]
 [0.54571277 0.4542873 ]
 [0.45948133 0.54051864]
 [0.6057914  0.39420864]
 [0.4465654  0.55343455]
 [0.60244924 0.39755073]
 [0.57046854 0.4295314 ]
 [0.6301627  0.3698373 ]
 [0.4920884  0.5079116 ]
 [0.4035052  0.5964948 ]
 [0.56033206 0.43966797]
 [0.5877249  0.41227505]
 [0.56644195 0.43355805]
 [0.40571567 0.59428436]
 [0.48935604 0.5106439 ]
 [0.52726674 0.47273326]
 [0.3934633  0.60653675]
 [0.48530552 0.5146944 ]
 [0.4542345  0.5457655 ]
 [0.60464644 0.39535362]
 [0.5187622  0.48123786]
 [0.5898149  0.41018507]
 [0.6575892  0.34241083]
 [0.5853153  0.41468465]
 [0.5554536  0.44454637]
 [0.59884906 0.40115097]
 [0.5957013  0.40429872]
 [0.5611944  0.43880558]
 [0.6112777  0.3887223 ]]
```