In [2]:
```
# importing the necessary package
import pandas as pd
import numpy as np
```

In [3]:
```
# assigning the .csv file to a variable.
label=pd.read_csv("/Users/rohitbohra/Documents/aptos2019-blindness-det
```

In [4]:
```
# first 5 rows in the dataframe.
label.head()
```

Out[4]:

	Unnamed: 0	id_code	diagnosis
0	0	000c1434d8d7.png	2
1	1	00a8624548a9.png	2
2	2	00cb6555d108.png	1
3	3	0104b032c141.png	3
4	4	0124dffecf29.png	1

In [8]:
```
# column of the datasets
list(label.columns)
```

Out[8]:
```
['Unnamed: 0', 'id_code', 'diagnosis']
```

In [10]:
```
# dropping the unnecessary column from the dataframe
label=label.drop(['Unnamed: 0'], axis=1)
list(label.columns)
```

Out[10]:
```
['id_code', 'diagnosis']
```

In [11]:
```
# first 5 rows of data
label.head()
```

Out[11]:

	id_code	diagnosis
0	000c1434d8d7.png	2
1	00a8624548a9.png	2
2	00cb6555d108.png	1
3	0104b032c141.png	3
4	0124dffecf29.png	1

In [14]:
```
#importing the required packages
import cv2
import os

path="/Users/rohitbohra/Documents/aptos2019-blindness-detection/train_

#functiion for resizing the images
def get_image(path, id_code, size):
    img_path = os.path.join(path, id_code)
    image = cv2.imread(img_path)
    image = cv2.resize(image, (size,size))
    image_arr = image.reshape( size,size, 3).astype('float32')/255
    #print(image_arr.shape)
    return image_arr
```

In [15]:
```
# reshaping the image shape
all_images = []
all_images.append(label['id_code'].apply(lambda code: get_image('/User
x_train = np.array(all_images)

x_train = x_train.reshape(x_train.shape[1], 96,96, 3).astype('float32'
```

In [17]:
```
# labels of the dataset to be assigned to a variable
y_train = label['diagnosis']
```

In [18]:
```
# converting the label into categorical data
from keras.utils import np_utils
y_train = np_utils.to_categorical(y_train)

# number of classes
num_classes = y_train.shape[1]
```

```
/Users/rohitbohra/anaconda3/lib/python3.6/site-packages/h5py/__init_
_.py:36: FutureWarning: Conversion of the second argument of issubdt
ype from `float` to `np.floating` is deprecated. In future, it will
be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [19]:
```
# splitting of dataset into train and test data with 80 - 20 split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_train, y_train,
```

In [21]:
```
# checking the shape of the train data and test data
print(X_train.shape)
print(X_test.shape)
```

```
(743, 96, 96, 3)
(186, 96, 96, 3)
```

In [22]:
```
# importing the packages for Neural Network uisng keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout, GaussianNoise, GaussianDropout
from keras.layers import Flatten, BatchNormalization
from keras.layers.convolutional import Conv2D, SeparableConv2D

from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras import backend as K
from keras import regularizers
```

In [23]:
```
# building a function with the required layers and regularization laye

def build_model():
    # create model
    model = Sequential()
    model.add(Conv2D(15, (9, 9), input_shape=[96,96,3], activation='re
    model.add(GaussianDropout(0.3))
    model.add(Conv2D(30, (7, 7), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(30, (7, 7), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(50, (7, 7), activation='relu'))
    model.add(Conv2D(50, (5, 5), activation='relu'))

    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(256, activation='relu', kernel_regularizer=regular
    model.add(Dense(128, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(num_classes, activation='softmax', kernel_regulari
                    ,activity_regularizer=regularizers.l1(0.01)))

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', m
    return model
```

In [24]:
```
model = build_model()
```

In [25]:
```
# using early stopping which will automatically stop the training of d

from keras.callbacks import EarlyStopping, ModelCheckpoint
es= EarlyStopping(monitor='val_loss', mode ='min', verbose = 1, patien
mc = ModelCheckpoint('model.h5', monitor='val_loss', save_best_only = 
```

In [26]:
```
# Fitting the model and checking how the model performs
model.fit(X_train, y_train, validation_data = (X_test, y_test),epochs=
```

```
Train on 743 samples, validate on 186 samples
Epoch 1/1000
743/743 [==============================] - 62s 84ms/step - loss: 4.4
149 - acc: 0.5047 - val_loss: 2.8607 - val_acc: 0.4785

Epoch 00001: val_loss improved from inf to 2.86069, saving model to
model.h5
Epoch 2/1000
743/743 [==============================] - 62s 84ms/step - loss: 2.3
953 - acc: 0.5087 - val_loss: 1.9695 - val_acc: 0.4785

Epoch 00002: val_loss improved from 2.86069 to 1.96951, saving model
to model.h5
Epoch 3/1000
743/743 [==============================] - 60s 80ms/step - loss: 1.8
223 - acc: 0.5087 - val_loss: 1.7015 - val_acc: 0.4785

Epoch 00003: val_loss improved from 1.96951 to 1.70146, saving model
to model.h5
```

In [28]:
```
history=model.fit(X_train, y_train, validation_data = (X_test, y_test)
```

```
Train on 743 samples, validate on 186 samples
Epoch 1/20
743/743 [==============================] - 60s 80ms/step - loss: 1.1
116 - acc: 0.7254 - val_loss: 1.1779 - val_acc: 0.6828

Epoch 00001: val_loss did not improve from 1.17067
Epoch 2/20
743/743 [==============================] - 60s 80ms/step - loss: 1.1
092 - acc: 0.7079 - val_loss: 1.1991 - val_acc: 0.7151

Epoch 00002: val_loss did not improve from 1.17067
Epoch 3/20
743/743 [==============================] - 59s 80ms/step - loss: 1.1
018 - acc: 0.7201 - val_loss: 1.2092 - val_acc: 0.7204

Epoch 00003: val_loss did not improve from 1.17067
Epoch 4/20
743/743 [==============================] - 59s 80ms/step - loss: 1.0
971 - acc: 0.7227 - val_loss: 1.2166 - val_acc: 0.7204

Epoch 00004: val_loss did not improve from 1.17067
Epoch 5/20
743/743 [==============================] - 59s 80ms/step - loss: 1.0
815 - acc: 0.7322 - val_loss: 1.2140 - val_acc: 0.6882

Epoch 00005: val_loss did not improve from 1.17067
Epoch 6/20
743/743 [==============================] - 59s 80ms/step - loss: 1.0
```

```
977 - acc: 0.7281 - val_loss: 1.2458 - val_acc: 0.6989

Epoch 00006: val_loss did not improve from 1.17067
Epoch 7/20
743/743 [==============================] - 60s 80ms/step - loss: 1.1
037 - acc: 0.7362 - val_loss: 1.1983 - val_acc: 0.7043

Epoch 00007: val_loss did not improve from 1.17067
Epoch 8/20
743/743 [==============================] - 59s 80ms/step - loss: 1.1
071 - acc: 0.7268 - val_loss: 1.2300 - val_acc: 0.7473

Epoch 00008: val_loss did not improve from 1.17067
Epoch 9/20
743/743 [==============================] - 60s 80ms/step - loss: 1.1
573 - acc: 0.7079 - val_loss: 1.2441 - val_acc: 0.7151

Epoch 00009: val_loss did not improve from 1.17067
Epoch 10/20
743/743 [==============================] - 59s 80ms/step - loss: 1.1
431 - acc: 0.7026 - val_loss: 1.2414 - val_acc: 0.7204

Epoch 00010: val_loss did not improve from 1.17067
Epoch 11/20
743/743 [==============================] - 67s 90ms/step - loss: 1.0
992 - acc: 0.7174 - val_loss: 1.1559 - val_acc: 0.7204

Epoch 00011: val_loss improved from 1.17067 to 1.15591, saving model
to model.h5
Epoch 12/20
743/743 [==============================] - 68s 91ms/step - loss: 1.0
830 - acc: 0.7416 - val_loss: 1.1671 - val_acc: 0.6989

Epoch 00012: val_loss did not improve from 1.15591
Epoch 13/20
743/743 [==============================] - 61s 83ms/step - loss: 1.0
678 - acc: 0.7443 - val_loss: 1.1838 - val_acc: 0.7097

Epoch 00013: val_loss did not improve from 1.15591
Epoch 14/20
743/743 [==============================] - 59s 79ms/step - loss: 1.0
760 - acc: 0.7281 - val_loss: 1.2080 - val_acc: 0.7097

Epoch 00014: val_loss did not improve from 1.15591
Epoch 15/20
743/743 [==============================] - 59s 79ms/step - loss: 1.0
732 - acc: 0.7214 - val_loss: 1.2278 - val_acc: 0.6935

Epoch 00015: val_loss did not improve from 1.15591
Epoch 16/20
743/743 [==============================] - 59s 80ms/step - loss: 1.0
728 - acc: 0.7429 - val_loss: 1.2052 - val_acc: 0.7043
```

```
Epoch 00016: val_loss did not improve from 1.15591
Epoch 17/20
743/743 [==============================] - 59s 79ms/step - loss: 1.0
690 - acc: 0.7281 - val_loss: 1.3196 - val_acc: 0.6828

Epoch 00017: val_loss did not improve from 1.15591
Epoch 18/20
743/743 [==============================] - 59s 80ms/step - loss: 1.0
978 - acc: 0.7254 - val_loss: 1.2295 - val_acc: 0.6935

Epoch 00018: val_loss did not improve from 1.15591
Epoch 19/20
743/743 [==============================] - 59s 79ms/step - loss: 1.0
706 - acc: 0.7389 - val_loss: 1.2184 - val_acc: 0.7097

Epoch 00019: val_loss did not improve from 1.15591
Epoch 20/20
743/743 [==============================] - 60s 81ms/step - loss: 1.0
509 - acc: 0.7402 - val_loss: 1.2059 - val_acc: 0.6882

Epoch 00020: val_loss did not improve from 1.15591
```

In [29]:

```
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_dynamic_val(x, vy, ty, ax, colors=['b']):
    ax.plot(x, vy, 'b', label="Validation Accuracy")
    ax.plot(x, ty, 'r', label="Train Accuracy")
    plt.legend()
    plt.grid(True)
    fig.canvas.draw()
```

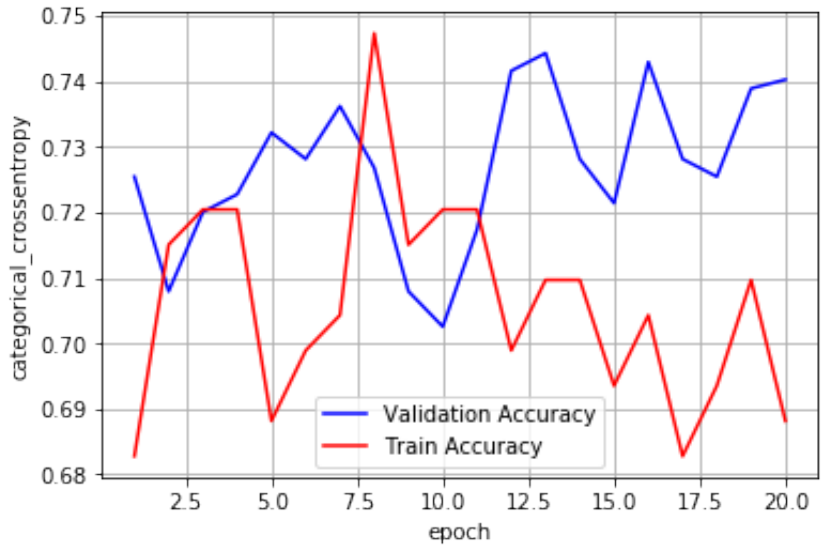
In [32]:
```
# Checking the accuracy of model on test data
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy:', score[1])

fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('categorical_crossentropy')

# list of epoch numbers
x = list(range(1,21))

vy = history.history['acc']
ty = history.history['val_acc']
plt_dynamic_val(x, vy, ty, ax)
```

```
Test Accuracy: 0.6881720423698425
```



In [46]:
```
def build_model1():
    # create model
    model = Sequential()
    model.add(Conv2D(250, (9, 9), input_shape=[96,96,3], activation='r
    model.add(Dropout(0.5))
    model.add(Conv2D(230, (7, 7), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.5))
    model.add(Conv2D(150, (7, 7), activation='relu'))
    model.add(Conv2D(130, (5, 5), activation='relu'))

    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(526, activation='relu', kernel_regularizer=regular
    model.add(Dense(254, activation='relu'))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(50, activation='relu'))
    model.add(Dense(num_classes, activation='softmax', kernel_regulari
                    ,activity_regularizer=regularizers.l1(0.01)))
    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='adam', m
    return model
```

In [47]:
```
model = build_model1()
```

In [48]:
```
from keras.callbacks import EarlyStopping, ModelCheckpoint
es= EarlyStopping(monitor='val_loss', mode ='min', verbose = 1, patien
mc = ModelCheckpoint('model1.h5', monitor='val_loss', save_best_only =
```

In [49]:
```
model.fit(X_train, y_train, validation_data = (X_test, y_test),epochs=
```

```
Epoch 34/1000
743/743 [==============================] - 2028s 3s/step - loss: 1.1
105 - acc: 0.7133 - val_loss: 1.1684 - val_acc: 0.6935

Epoch 00034: val_loss did not improve from 1.16347
Epoch 35/1000
743/743 [==============================] - 2031s 3s/step - loss: 1.1
072 - acc: 0.6878 - val_loss: 1.1776 - val_acc: 0.6720

Epoch 00035: val_loss did not improve from 1.16347
Epoch 36/1000
743/743 [==============================] - 2034s 3s/step - loss: 1.0
836 - acc: 0.7133 - val_loss: 1.1877 - val_acc: 0.6882

Epoch 00036: val_loss did not improve from 1.16347
Epoch 37/1000
743/743 [==============================] - 2029s 3s/step - loss: 1.1
097 - acc: 0.7174 - val_loss: 1.2903 - val_acc: 0.6882

Epoch 00037: val_loss did not improve from 1.16347
```

In [57]:
```
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy:', score[1])
```

```
Test Accuracy: 0.47849462525818937
```

In [58]:
```
from prettytable import PrettyTable
x = PrettyTable()

x.field_names = ["No. of layers", "Train Accuracy","Train loss","Test a
x.add_row([14,74.02,1.0509,68.82,1.2059])
x.add_row([14,71.87,1.1255,70.97,1.1993])


print('\t\t\t\tPerformance Table')
print(x)
```

```
                                Performance Table
+---------------+----------------+------------+---------------+-----
------+
| No. of layers | Train Accuracy | Train loss | Test accuracy | Test
loss |
+---------------+----------------+------------+---------------+-----
------+
|       14      |     74.02      |   1.0509   |     68.82     |   1.
2059  |
|       14      |     71.87      |   1.1255   |     70.97     |   1.
1993  |
+---------------+----------------+------------+---------------+-----
------+
```

Conclusion

1) The data is imbalance, the count of each class is highly imbalance.

2) The image size is reshaped.

3) We build a neural network model with a total of 14 layers and use regularization to avoid overfitting.

4) We use EarlyStopping to automatically stop the training of the model when the model is not imporving.

5) The first model's train loss = 1.0509 and train accuracy = 74.02, and test loss = 1.2059 and test accuracy = 68.82

6) The model is slightly overfit as the model performs well for training dataset and not for testing dstaset.

7) The second model's train loss = 1.1255 and train accuracy = 71.87, and test loss = 1.1993 and test accuracy = 70.97

8) The model is a good fit for both training dataset and testing dataset.

Future Enhancement

1) For model to perform better, we can use other modeling techniques like transform learning.

2) As the dataset is very less to build a strong model, we can use data generator to generate new data from existing dataset.