Convolutional Neural Network

In [0]:

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/au
th?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.goog
leusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&r
esponse_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fa
uth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20
https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20ht
tps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

In [0]:

```python
import pandas as pd
import numpy as np
```

In [0]:

```python
import matplotlib.pyplot as plt
```

In [0]:

```python
df3 = pd.read_csv("/content/drive/My Drive/image classification project/df3_final.csv",header=None) #importing data from directory
X = df3.iloc[:,7:3079].values
y = df3.iloc[:, 3079].values
np.random.seed(3)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,random_state = 0)
```

In [0]:

```python
X_train=X_train/255.0
X_test=X_test/255.0
X_train=X_train.reshape((X_train.shape[0]),32,32,3)            #resizeing of the data
X_test=X_test.reshape((X_test.shape[0]),32,32,3)
```

In [0]:

```python
X_test.shape
```

Out[0]:

```
(2003, 32, 32, 3)
```

In [0]:

```
#importing all important libraries
import keras
from keras.utils.np_utils import to_categorical # used for converting labels to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
import itertools
from keras.layers.normalization import BatchNormalization
from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.optimizers import Adam
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade (https://www.tensorflow.org/guide/migrate) now or ensure your notebook will continue to use TensorFlow 1.x via the `%tensorflow_version 1.x` magic: more info (https://colab.research.google.com/notebooks/tensorflow_version.ipynb).

In [ ]:

```
# Encode categorical features as a one-hot numeric array
```

In [0]:

```
from keras.utils import to_categorical
y_train_one_hot = to_categorical(y_train)
y_test_one_hot = to_categorical(y_test)
```

In [0]:

```
input_shape = (32,32, 3)
num_classes = 7

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',padding = 'Same',input_shape=input_shape))

model.add(MaxPool2D(pool_size = (2, 2)))
model.add(Dropout(0.25))



model.add(Conv2D(128, (3, 3), activation='relu',padding = 'Same'))

model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_19 (Conv2D)           (None, 32, 32, 32)        896       
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 16, 16, 32)        0         
_________________________________________________________________
dropout_28 (Dropout)         (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 16, 16, 128)       36992     
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 8, 8, 128)         0         
_________________________________________________________________
dropout_29 (Dropout)         (None, 8, 8, 128)         0         
_________________________________________________________________
flatten_10 (Flatten)         (None, 8192)              0         
_________________________________________________________________
dense_19 (Dense)             (None, 256)               2097408   
_________________________________________________________________
dropout_30 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_20 (Dense)             (None, 7)                 1799      
=================================================================
Total params: 2,137,095
Trainable params: 2,137,095
Non-trainable params: 0
_________________________________________________________________
```

In [0]:

```
model.compile(optimizer = 'adam' , loss = "categorical_crossentropy",
metrics=["accuracy"])

epochs = 50
batch_size = 10
history = model.fit(X_train,y_train_one_hot, batch_size=batch_size,
                              epochs = epochs, validation_split=0.1)
```

```
Train on 7210 samples, validate on 802 samples
Epoch 1/50
7210/7210 [==============================] - 5s 672us/step - loss:
0.9964 - acc: 0.6660 - val_loss: 0.9458 - val_acc: 0.6820
Epoch 2/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.9078 - acc: 0.6730 - val_loss: 0.8989 - val_acc: 0.6858
Epoch 3/50
7210/7210 [==============================] - 4s 512us/step - loss:
0.8776 - acc: 0.6843 - val_loss: 0.7844 - val_acc: 0.7045
Epoch 4/50
7210/7210 [==============================] - 4s 507us/step - loss:
0.8367 - acc: 0.6946 - val_loss: 0.7701 - val_acc: 0.7195
Epoch 5/50
7210/7210 [==============================] - 4s 510us/step - loss:
0.7954 - acc: 0.7026 - val_loss: 0.7554 - val_acc: 0.7232
Epoch 6/50
7210/7210 [==============================] - 4s 511us/step - loss:
```

```
0.7691 - acc: 0.7100 - val_loss: 0.7213 - val_acc: 0.7257
Epoch 7/50
7210/7210 [==============================] - 4s 508us/step - loss:
0.7519 - acc: 0.7189 - val_loss: 0.7262 - val_acc: 0.7382
Epoch 8/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.7389 - acc: 0.7225 - val_loss: 0.7263 - val_acc: 0.7406
Epoch 9/50
7210/7210 [==============================] - 4s 504us/step - loss:
0.7153 - acc: 0.7287 - val_loss: 0.6772 - val_acc: 0.7431
Epoch 10/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.6989 - acc: 0.7408 - val_loss: 0.6917 - val_acc: 0.7531
Epoch 11/50
7210/7210 [==============================] - 4s 507us/step - loss:
0.6945 - acc: 0.7398 - val_loss: 0.7364 - val_acc: 0.7544
Epoch 12/50
7210/7210 [==============================] - 4s 504us/step - loss:
0.6765 - acc: 0.7481 - val_loss: 0.6676 - val_acc: 0.7581
Epoch 13/50
7210/7210 [==============================] - 4s 511us/step - loss:
0.6621 - acc: 0.7452 - val_loss: 0.6709 - val_acc: 0.7681
Epoch 14/50
7210/7210 [==============================] - 4s 507us/step - loss:
0.6554 - acc: 0.7552 - val_loss: 0.7361 - val_acc: 0.7382
Epoch 15/50
7210/7210 [==============================] - 4s 502us/step - loss:
0.6437 - acc: 0.7614 - val_loss: 0.6756 - val_acc: 0.7594
Epoch 16/50
7210/7210 [==============================] - 4s 512us/step - loss:
0.6292 - acc: 0.7580 - val_loss: 0.6855 - val_acc: 0.7494
Epoch 17/50
7210/7210 [==============================] - 4s 515us/step - loss:
0.6176 - acc: 0.7739 - val_loss: 0.6757 - val_acc: 0.7469
Epoch 18/50
7210/7210 [==============================] - 4s 508us/step - loss:
0.5986 - acc: 0.7731 - val_loss: 0.7106 - val_acc: 0.7631
Epoch 19/50
7210/7210 [==============================] - 4s 508us/step - loss:
0.6025 - acc: 0.7786 - val_loss: 0.6677 - val_acc: 0.7718
Epoch 20/50
7210/7210 [==============================] - 4s 502us/step - loss:
0.5911 - acc: 0.7796 - val_loss: 0.6985 - val_acc: 0.7606
Epoch 21/50
7210/7210 [==============================] - 4s 512us/step - loss:
0.5772 - acc: 0.7813 - val_loss: 0.6592 - val_acc: 0.7569
Epoch 22/50
7210/7210 [==============================] - 4s 526us/step - loss:
0.5590 - acc: 0.7836 - val_loss: 0.6514 - val_acc: 0.7631
Epoch 23/50
7210/7210 [==============================] - 4s 511us/step - loss:
0.5650 - acc: 0.7904 - val_loss: 0.6657 - val_acc: 0.7643
Epoch 24/50
7210/7210 [==============================] - 4s 508us/step - loss:
0.5339 - acc: 0.7974 - val_loss: 0.6758 - val_acc: 0.7756
Epoch 25/50
```

```
7210/7210 [==============================] - 4s 513us/step - loss:
0.5333 - acc: 0.7939 - val_loss: 0.6799 - val_acc: 0.7830
Epoch 26/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.5249 - acc: 0.7960 - val_loss: 0.6740 - val_acc: 0.7618
Epoch 27/50
7210/7210 [==============================] - 4s 502us/step - loss:
0.5195 - acc: 0.8039 - val_loss: 0.7030 - val_acc: 0.7656
Epoch 28/50
7210/7210 [==============================] - 4s 512us/step - loss:
0.5059 - acc: 0.8097 - val_loss: 0.6850 - val_acc: 0.7544
Epoch 29/50
7210/7210 [==============================] - 4s 500us/step - loss:
0.5024 - acc: 0.8024 - val_loss: 0.6987 - val_acc: 0.7581
Epoch 30/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.4862 - acc: 0.8144 - val_loss: 0.7131 - val_acc: 0.7668
Epoch 31/50
7210/7210 [==============================] - 4s 505us/step - loss:
0.4817 - acc: 0.8178 - val_loss: 0.7249 - val_acc: 0.7519
Epoch 32/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.4760 - acc: 0.8179 - val_loss: 0.6907 - val_acc: 0.7743
Epoch 33/50
7210/7210 [==============================] - 4s 514us/step - loss:
0.4528 - acc: 0.8252 - val_loss: 0.6877 - val_acc: 0.7706
Epoch 34/50
7210/7210 [==============================] - 4s 509us/step - loss:
0.4597 - acc: 0.8252 - val_loss: 0.7114 - val_acc: 0.7656
Epoch 35/50
7210/7210 [==============================] - 4s 521us/step - loss:
0.4485 - acc: 0.8302 - val_loss: 0.6720 - val_acc: 0.7718
Epoch 36/50
7210/7210 [==============================] - 4s 520us/step - loss:
0.4392 - acc: 0.8318 - val_loss: 0.7784 - val_acc: 0.7581
Epoch 37/50
7210/7210 [==============================] - 4s 502us/step - loss:
0.4457 - acc: 0.8344 - val_loss: 0.6899 - val_acc: 0.7656
Epoch 38/50
7210/7210 [==============================] - 4s 504us/step - loss:
0.4266 - acc: 0.8395 - val_loss: 0.7121 - val_acc: 0.7656
Epoch 39/50
7210/7210 [==============================] - 4s 511us/step - loss:
0.4067 - acc: 0.8442 - val_loss: 0.8055 - val_acc: 0.7406
Epoch 40/50
7210/7210 [==============================] - 4s 506us/step - loss:
0.4166 - acc: 0.8405 - val_loss: 0.7133 - val_acc: 0.7618
Epoch 41/50
7210/7210 [==============================] - 4s 501us/step - loss:
0.4139 - acc: 0.8413 - val_loss: 0.7072 - val_acc: 0.7706
Epoch 42/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.4124 - acc: 0.8402 - val_loss: 0.7207 - val_acc: 0.7731
Epoch 43/50
7210/7210 [==============================] - 4s 510us/step - loss:
0.4011 - acc: 0.8469 - val_loss: 0.7575 - val_acc: 0.7656
```

```
Epoch 44/50
7210/7210 [==============================] - 4s 510us/step - loss:
0.3987 - acc: 0.8479 - val_loss: 0.7013 - val_acc: 0.7656
Epoch 45/50
7210/7210 [==============================] - 4s 503us/step - loss:
0.3824 - acc: 0.8542 - val_loss: 0.6898 - val_acc: 0.7631
Epoch 46/50
7210/7210 [==============================] - 4s 502us/step - loss:
0.3890 - acc: 0.8533 - val_loss: 0.8221 - val_acc: 0.7494
Epoch 47/50
7210/7210 [==============================] - 4s 507us/step - loss:
0.3726 - acc: 0.8578 - val_loss: 0.7394 - val_acc: 0.7643
Epoch 48/50
7210/7210 [==============================] - 4s 509us/step - loss:
0.3767 - acc: 0.8556 - val_loss: 0.7697 - val_acc: 0.7643
Epoch 49/50
7210/7210 [==============================] - 4s 504us/step - loss:
0.3677 - acc: 0.8578 - val_loss: 0.7900 - val_acc: 0.7693
Epoch 50/50
7210/7210 [==============================] - 4s 504us/step - loss:
0.3576 - acc: 0.8662 - val_loss: 0.7340 - val_acc: 0.7656
```

In [0]:

```
model.evaluate(X_test, y_test_one_hot)[1]
```

```
2003/2003 [==============================] - 0s 73us/step
```

Out[0]:

```
0.7553669494268481
```

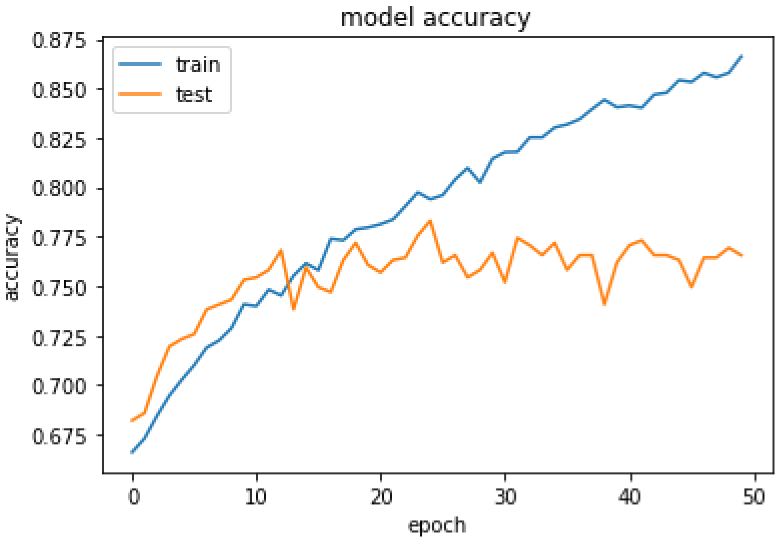
In [0]:

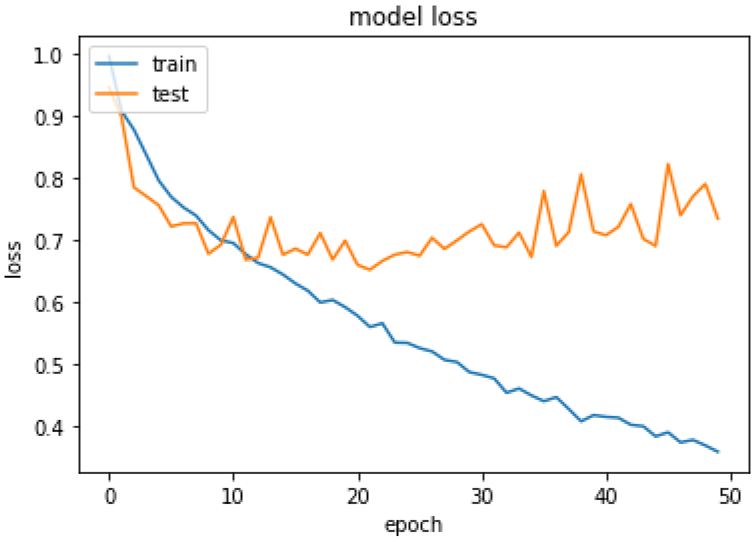
```
model.save_weights("model.h5") #savinf Model's weight into model.h5
```

Plots of Training and validation accuracy and loss

In [0]:

```
# Summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Confusion Matrix and Classification reports

```
In [0]: #Prediction of Testing image
        preds=np.round(model.predict(X_test),0)
```

```
In [0]: from sklearn.metrics import classification_report, confusion_matrix
```

```
In [0]: cm = confusion_matrix(y_test, preds.argmax(axis=1))
```

In [0]:

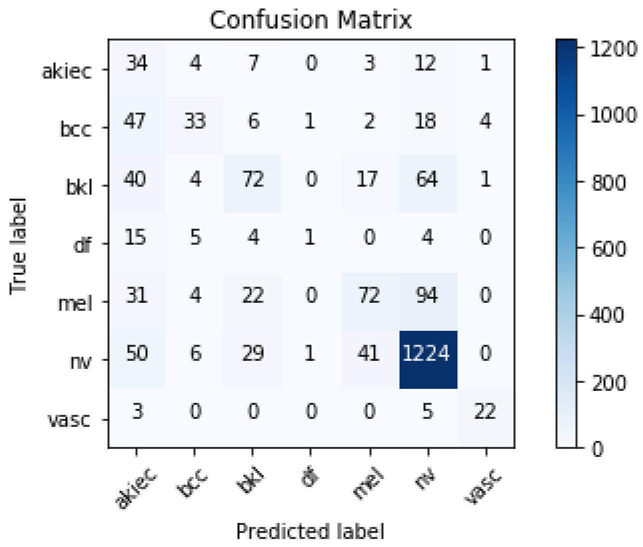
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    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)

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

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

In [0]:

```
cm_plot_labels = ['akiec', 'bcc', 'bkl', 'df', 'mel','nv', 'vasc']
```

```
In [0]: plot_confusion_matrix(cm, cm_plot_labels, title='Confusion Matrix')
```

```
Confusion matrix, without normalization
[[  34    4    7    0    3   12    1]
 [  47   33    6    1    2   18    4]
 [  40    4   72    0   17   64    1]
 [  15    5    4    1    0    4    0]
 [  31    4   22    0   72   94    0]
 [  50    6   29    1   41 1224    0]
 [   3    0    0    0    0    5   22]]
```



```
In [0]: # Generate a classification report
        report = classification_report(y_test_one_hot, preds, target_names=c
        m_plot_labels)
        print(report)
```

```
              precision    recall  f1-score   support

       akiec       0.52      0.18      0.27        61
         bcc       0.59      0.30      0.40       111
         bkl       0.51      0.36      0.43       198
          df       0.33      0.03      0.06        29
         mel       0.53      0.32      0.40       223
          nv       0.86      0.91      0.88      1351
        vasc       0.79      0.73      0.76        30

   micro avg       0.80      0.72      0.75      2003
   macro avg       0.59      0.41      0.46      2003
weighted avg       0.76      0.72      0.72      2003
 samples avg       0.72      0.72      0.72      2003


/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classificati
on.py:1272: UndefinedMetricWarning: Precision and F-score are ill-de
fined and being set to 0.0 in samples with no predicted labels. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [0]:

```
#Saving model
from google.colab import files
files.download('model.h5')
```