In [3]:

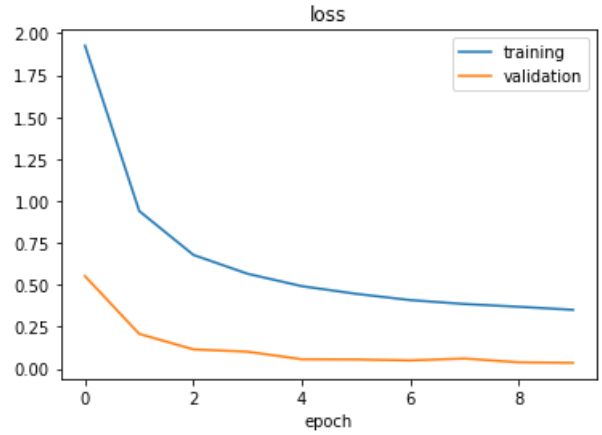
```
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import ImageDataGenerator



myData = "C:/Users/ASUS/Downloads/Deep learning/Szakdolgozat program/myData"
batch_size_val=20
steps_per_epoch_val=2000
epochs_val=10
imageDimension = (32,32)
testRate = 0.2
validationRate = 0.2




count = 0
images = []
classNo = []
myList = os.listdir(myData)
print("Total Classes Detected:",len(myList))
noOfClasses=len(myList)
for x in range (0,len(myList)):
    myPicList = os.listdir(myData+"/"+str(count))
    for y in myPicList:
        curImg = cv2.imread(myData+"/"+str(count)+"/"+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end =" ")
    count +=1
print(" ")
images = np.array(images)
classNo = np.array(classNo)

print(count)


X_train, X_test, y_train, y_test = train_test_split(images,
                                                    classNo,
                                                    test_size=testRate)

X_train, X_validation, y_train, y_validation = train_test_split(X_train,
                                                                y_train,
                                                                test_size=validationRate)




def bgr2gray(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img

def equalizeHistogram(img):
    img =cv2.equalizeHist(img)
    return img

def preprocess(img):
    img = bgr2gray(img)
```

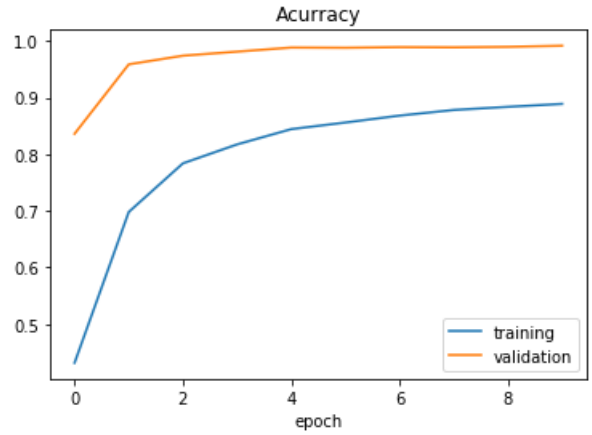
```
    img=equalizeHistogram(img)
    img = img/255
    return img



X_train=np.array(list(map(preprocess,X_train)))
X_validation=np.array(list(map(preprocess,X_validation)))
X_test=np.array(list(map(preprocess,X_test)))



X_train=X_train.reshape(X_train.shape[0],X_train.shape[1],X_train.shape[2],1)
X_validation=X_validation.reshape(X_validation.shape[0],X_validation.shape[1],X_validation.shape[2
],1)
X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],X_test.shape[2],1)




dataGen=ImageDataGenerator(width_shift_range=0.1,
                           height_shift_range=0.1,
                           zoom_range=0.2,
                           shear_range=0.1,
                           rotation_range=10)


dataGen.fit(X_train)









y_train = to_categorical(y_train,
                         noOfClasses)
y_validation = to_categorical(y_validation,
                              noOfClasses)
y_test = to_categorical(y_test,
                        noOfClasses)







def myModel():

    model= Sequential()
    model.add((Conv2D(60,(3,3),input_shape=(imageDimension[0],
                                            imageDimension[1],
                                            1),
                                       activation='relu')))
    model.add((Conv2D(60, (3,3), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))


    model.add((Conv2D(30, (3,3), activation='relu')))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(500,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses,activation='softmax'))

    model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
    return model


model = myModel()
print(model.summary())
history=model.fit_generator(dataGen.flow(X_train,
                                         y_train,
```

```
                                    batch_size=batch_size_val),
                            steps_per_epoch=steps_per_epoch_val,
                            epochs=epochs_val,
                            validation_data=(X_validation,
                                             y_validation))




plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training','validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
plt.show()
score =model.evaluate(X_test,y_test,verbose=0)
print('Test loss:',score[0])
print('Test Accuracy:',score[1])
```

```
Total Classes Detected: 43
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 3
6 37 38 39 40 41 42
43
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 30, 30, 60)        600
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 60)        32460
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 60)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 30)        16230
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 6, 6, 30)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 6, 6, 30)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 1080)              0
_________________________________________________________________
dense_5 (Dense)              (None, 500)               540500
_________________________________________________________________
dropout_6 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_6 (Dense)              (None, 43)                21543
=================================================================
Total params: 611,333
Trainable params: 611,333
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/10
2000/2000 [==============================] - 209s 104ms/step - loss: 1.9253 - accuracy: 0.4318 - v
al_loss: 0.5513 - val_accuracy: 0.8360
Epoch 2/10
2000/2000 [==============================] - 209s 104ms/step - loss: 0.9397 - accuracy: 0.6980 - v
al_loss: 0.2071 - val_accuracy: 0.9585
Epoch 3/10
2000/2000 [==============================] - 212s 106ms/step - loss: 0.6779 - accuracy: 0.7838 - v
al_loss: 0.1148 - val_accuracy: 0.9738
Epoch 4/10
2000/2000 [==============================] - 213s 106ms/step - loss: 0.5655 - accuracy: 0.8173 - v
al_loss: 0.1010 - val_accuracy: 0.9810
Epoch 5/10
2000/2000 [==============================] - 215s 107ms/step - loss: 0.4915 - accuracy: 0.8442 - v
al_loss: 0.0558 - val_accuracy: 0.9881
```

```
Epoch 6/10
2000/2000 [==============================] - 218s 109ms/step - loss: 0.4461 - accuracy: 0.8559 - v
al_loss: 0.0546 - val_accuracy: 0.9878
Epoch 7/10
2000/2000 [==============================] - 217s 108ms/step - loss: 0.4086 - accuracy: 0.8681 - v
al_loss: 0.0491 - val_accuracy: 0.9889
Epoch 8/10
2000/2000 [==============================] - 217s 108ms/step - loss: 0.3853 - accuracy: 0.8780 - v
al_loss: 0.0602 - val_accuracy: 0.9885
Epoch 9/10
2000/2000 [==============================] - 219s 109ms/step - loss: 0.3690 - accuracy: 0.8836 - v
al_loss: 0.0377 - val_accuracy: 0.9894
Epoch 10/10
2000/2000 [==============================] - 219s 109ms/step - loss: 0.3507 - accuracy: 0.8887 - v
al_loss: 0.0345 - val_accuracy: 0.9914
```






```
Test loss: 0.036753648272769036
Test Accuracy: 0.9916666746139526
```

In [ ]:

In [ ]:

In [ ]: