

Report Homework 2 – Machine Learning

Name: Shahkar Javid

Matricola: 2047305

Project Overview

1.1 Project Statement:

The goal is to classify to solve an image classification problem at your choice, following the requirements provided below.

Requirements on the problem:

The problem must be an image classification problem. Additional problems (such as segmentation, bounding boxes, etc.) are allowed as long as there is a base image classification problem.

Requirements on the dataset:

You can choose any dataset or subset of a data set with the following features:

- At least 10 classes
- At least 150 images per class

We have chosen two different datasets downloaded from Kaggle one is called food classification which contain 13434 images (training & testing) of 11 classes of different food : "Bread", "Dairy Product", "Dessert", "Egg", "Fried food", "Meat", "Noodles-Pasta", "Rice", "Seafood", "Soup", "Vegetable-Fruit".

The other dataset is called sign alphabet classification from which we will classify 26 alphabets in sign language

This dataset was downloaded from Kaggle into a computer from below link.

Food Classification: [Food-11 image dataset | Kaggle](#).

Sign Language: [Sign Language Gesture Images Dataset | Kaggle](#)

We implemented Convolutional neural network using different approaches change hyper parameters, changing models and loss functions to come as close as we can with the acceptable accuracy.

1.2 Preprocessing and loading:

The data set is provided with 11 folders of each classes with different amount of images. Initially we imported the "os" & "cv2" library to read and iterate over directories and fetch data as per our requirements. All the images are of high resolution and different sizes. So, I have resized all the images according to acceptable computation time.

Also, I have converted all the images from rgb into grayscale to lower the computation timing. All the images are then converted into arrays with their respective labels. We have also shuffled the training data so that model is feed by different class of image every time.

1.3 Splitting data into Train-Test:

After fetching data from the directories and converting them into Features(X) and Labels (Y) array, we have normalized the data by dividing it with 255. After that we have imported library from sklearn

learn i.e “`from` sklearn.model_selection `import` train_test_split” to split our features and label into training and testing data with the ration of 0.8 and 0.2 respectively.

Define CNN And Train It From Scratch For Image Classification

2.1 Model:

A CNN is composed of a hierarchy of levels. The input level is directly connected to the pixels of the image (input shape), the last levels are generally fully connected, while in the intermediate levels local connections and shared weights are used. A CNN-type neural network is created so that the various layers will be Convolutional. The most important thing is that in the first convolutional there is input shape which is in the size of the images considered plus the factor 3 because color is 'rgb'. For example in our case (60,60,1) is the input shape because we are working in gray scaling. In the convolutional layer it was decided to use the ReLU activation function which flattens the response to all negative values to zero, while leaving everything unchanged for values equal to or greater than zero. This simplicity, combined with the fact of drastically reducing the problem of vanishing gradient, so it is used here, in intermediate layers, where the quantity of steps and calculations is important. Calculating the derivative is in fact very simple: for all negative values it is equal to zero, while for the positive ones it is equal to 1. In the angled point in the origin the derivative is indefinite but is still set to zero by convention. Moreover after a convolutional layer it is convenient to insert a pooling to decrease the size of the matrix, because otherwise you would go to have a huge amount of data that you would not know how to manage. With MaxPooling we are going to reduce the size of the matrix, for example, after the second convolutional layer of our neural network we considered a MaxPooling of size (2x2) and the idea is that if we found something interesting in one of the four entrance cards that make up each square of the 2x2 grid, we'll just keep the bit more interesting. This reduces the size of a matrix while keeping the most important bits. In our case a MaxPooling is inserted after the second convolutional layer because for reasons related to performance, decreasing the size of the output of the convolutional and removing a MaxPooling the performance of our network increased in terms of accucy and loss function. The matrix of the last MaxPooling, which in my case is the third, is fed to another neural network, Fully Connected Neural Network. The important thing is that the last dense layer must have exactly the number of classes of your dataset as the unit number and that the activation function must be softmax because in this case we have a multiclass problem. As optimizer we had tried result with Adam and RMSProp was chosen with default learning rate as after a series of tests it was the one that provided the best performance given the small dataset. The model ends when the compile function is executed, in which it goes to configure the model which will then go to make the fit.

2.2 Metrics:

The metrics that we used for the classifications are the accuracy and the loss function. The chosen loss function is the **Sparse categorical crossentropy**, a logarithmic function that increases when the prediction is far from the label, so it must be kept as low as possible. It is used when our output is in integers instead of one-hot encoding.

```
Model: "sequential_5"
Layer (type)                 Output Shape              Param #
-----
conv2d_12 (Conv2D)           (None, 58, 58, 32)        320
max_pooling2d_12 (MaxPoolin (None, 29, 29, 32)         0
g2D)
conv2d_13 (Conv2D)           (None, 27, 27, 32)       9248
max_pooling2d_13 (MaxPoolin (None, 13, 13, 32)         0
g2D)
conv2d_14 (Conv2D)           (None, 11, 11, 32)       9248
max_pooling2d_14 (MaxPoolin (None, 5, 5, 32)          0
g2D)
flatten_5 (Flatten)          (None, 800)                0
dense_10 (Dense)              (None, 128)              102528
dense_11 (Dense)              (None, 64)                8256
dense_12 (Dense)              (None, 11)                 715
-----
Total params: 130,315
Trainable params: 130,315
Non-trainable params: 0
```

Fig 1 (Food Data Set)

2.3 Fit (Food Dataset):

As for the fit, the function `fit` is used, which trains the model on data fed in. It takes as input various parameters including the train set, the number of epochs (i.e. the number of iterations), steps per epochs (total number of steps to yield from generator before declaring one epoch finished and starting the next epoch.) It means that I can observe from the accuracy values, which are generated at each epoch, whether or not the overfitting problem arises. We have overfitting when the difference between train and test accuracy is very high. In our case, the number of epochs taken into consideration is 30-40. The more we increase the number of epochs, the accuracy will increase respectively.

```
train_y=np.array(train_y)

cnn.fit(train_X,train_y,epochs=40)
```

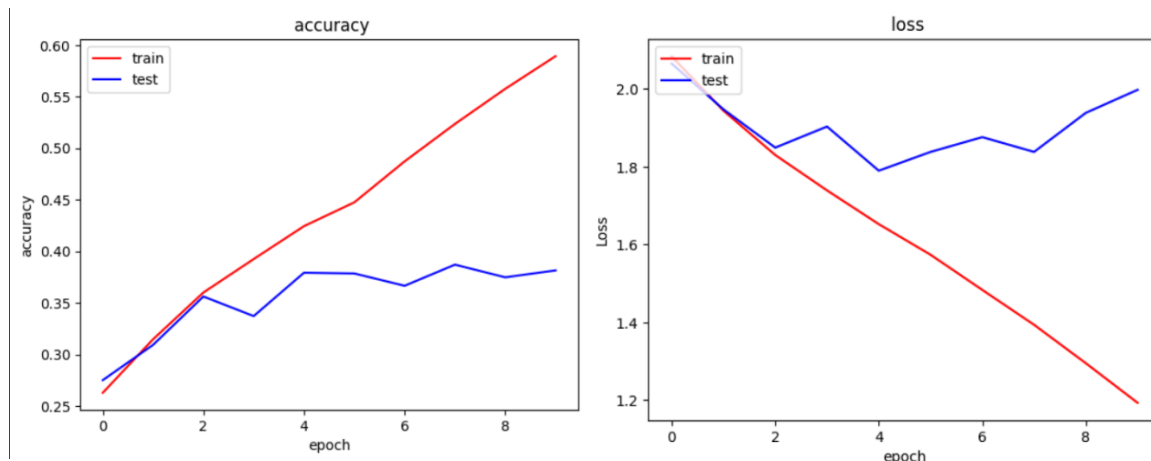
Fig 2

2.4 Evaluation:

The evaluation of the model was done using the classification report in which the values of precision, recall, F-score and accuracy are reported. Furthermore, the behavior of the model with the train is better analyzed and with the test it comes through the plt library. Also to better observe the presence of overfitting. We observed the food data set with different optimizer and with different learning rate yet due to small dataset we were unable to get expected result. Below it can be observe the accuracy and loss relationship between our train and test data that we have plotted with matplotlib library. Only **10 epochs** were considered for plotting to reduce computational time.

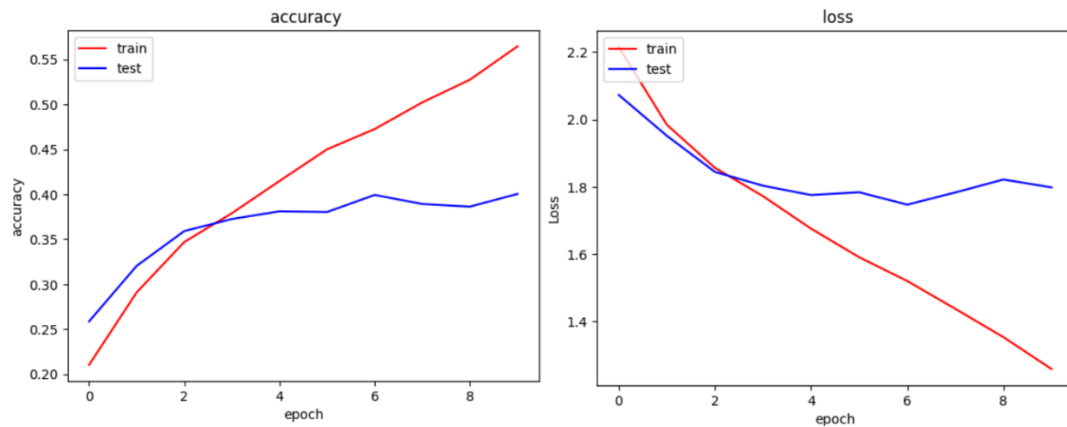
we observed from the below plotting of loss and accuracy against small number of epoch for train and test data; RMSProp and Adam optimizer worked relatively better than Adagrad. So we will try to maximize our accuracy considering RMSprop or Adam.

RMSProp:



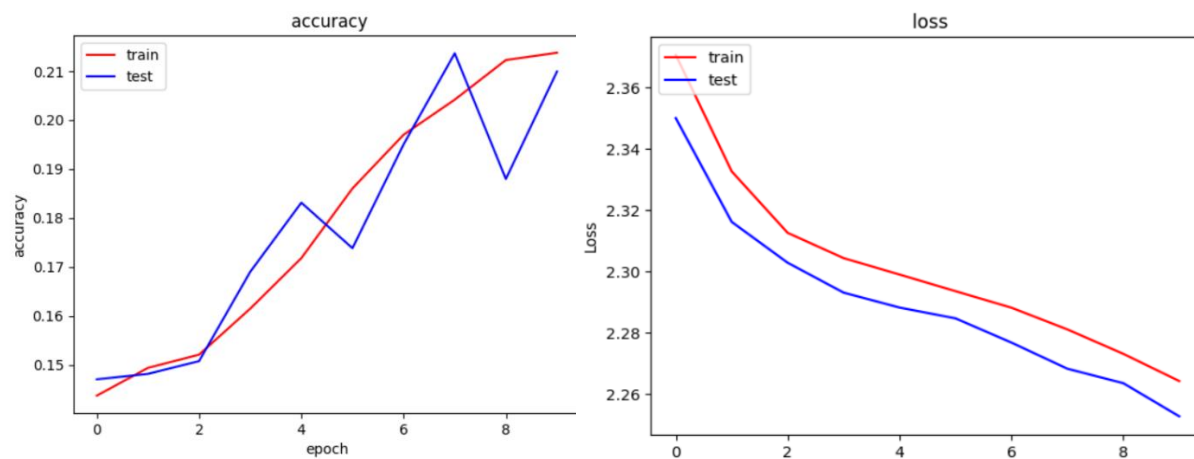
	precision	recall	f1-score	support
Bread	0.18	0.19	0.19	279
Dairy Product	0.17	0.19	0.18	159
Dessert	0.33	0.30	0.32	421
Egg	0.26	0.36	0.30	247
Fried food	0.33	0.39	0.36	213
Meat	0.47	0.38	0.42	328
Noodles-Pasta	0.58	0.40	0.48	141
Rice	0.33	0.35	0.34	84
Seafood	0.26	0.27	0.27	229
Soup	0.63	0.65	0.64	405
Vegetable-Fruit	0.39	0.28	0.33	181
accuracy			0.36	2687
macro avg	0.36	0.34	0.35	2687
weighted avg	0.37	0.36	0.36	2687

Adam:



	precision	recall	f1-score	support
Bread	0.25	0.36	0.29	279
Dairy Product	0.26	0.12	0.16	166
Dessert	0.30	0.33	0.32	395
Egg	0.36	0.36	0.36	292
Fried food	0.36	0.31	0.33	213
Meat	0.53	0.46	0.49	387
Noodles-Pasta	0.60	0.44	0.51	117
Rice	0.32	0.44	0.37	66
Seafood	0.30	0.30	0.30	223
Soup	0.64	0.74	0.69	385
Vegetable-Fruit	0.40	0.24	0.30	164
accuracy			0.40	2687
macro avg	0.39	0.37	0.38	2687
weighted avg	0.40	0.40	0.40	2687

Adagrad:



We have also introduced a batch size of 32 to see the effect on accuracy but unfortunately due to a small dataset we were unable to gain maximum accuracy. The accuracy we get in 10 epochs is around 40 percent; despite trying several learning rates and approaches including increasing the dense layers but accuracy didn't improve any significantly. To see if our model is working fine we will also fit this model on a large dataset.

2.5 Fit (Sign Language Dataset):

So now we will apply the same code and algorithm on a different data set relatively larger than the previous one called Sign alphabet recognition downloaded from Kaggle from the below given link.

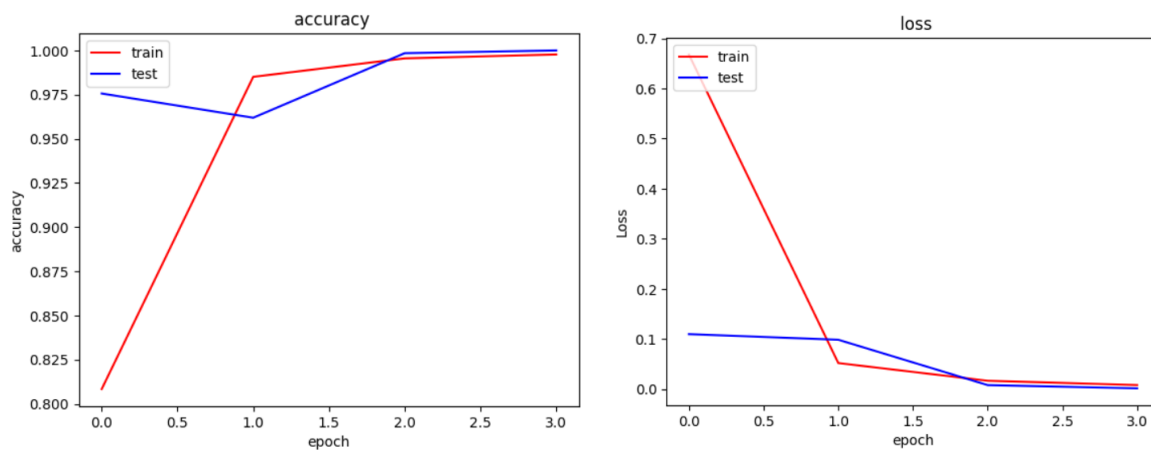
[Sign Language Gesture Images Dataset | Kaggle](#)



This data set contains 39,000 images of different alphabets which is then split 80 percent for training and 20 percent for testing. We have used the same model that we used previously for the food dataset; the only difference is that the output layers now have only 26 outputs with the Adam optimizer and default learning rates.

2.4 Evaluation:

So after fitting the data set on the same model we got surprisingly good results with an accuracy of nearly 99 percent both in training and testing after 3 epochs. This data set is also manageable with gray scaling because not too much detail is involved unlike the food data set which contains a lot of different details in the picture.



Output and classification report:

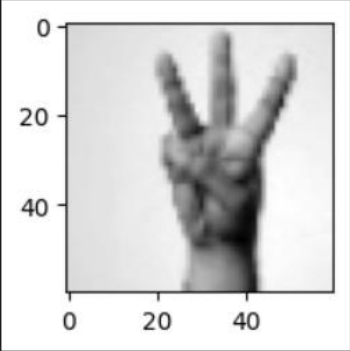
			precision	recall	f1-score	support
		A	0.98	1.00	0.99	311
		B	1.00	1.00	1.00	305
		C	0.99	0.99	0.99	276
		D	0.98	0.95	0.96	302
		E	1.00	1.00	1.00	292
		F	0.96	0.99	0.97	323
		G	1.00	0.91	0.95	293
		H	0.93	0.98	0.95	301
		I	0.99	0.98	0.98	280
		J	0.99	0.99	0.99	291
		K	0.99	0.97	0.98	303
		L	0.90	0.99	0.94	285
		M	1.00	1.00	1.00	298
		N	1.00	0.98	0.99	310
		O	1.00	1.00	1.00	347
		P	1.00	1.00	1.00	297
		Q	0.98	1.00	0.99	308
		R	0.95	0.91	0.93	318
		S	1.00	1.00	1.00	305
		T	1.00	0.99	0.99	309
		U	0.91	0.94	0.93	303
		V	0.99	0.96	0.97	309
		W	0.96	0.98	0.97	284
		X	1.00	0.99	1.00	307
		Y	1.00	1.00	1.00	275
		Z	0.99	1.00	0.99	268
		accuracy			0.98	7800

```
# print(y_pred)
output= np.argmax(y_pred)
print("output is : " + CATEGORIES[output])
```

✓ 0.1s

1/1 [=====] - 0s 23ms/step

output is : W



2 Conclusion:

So by evaluating our CNN model that we design we can say that it worked better when we have relatively larger dataset. Also we were doing all our training in gray scaling rather than RGB to reduce our computational time, which may have also lead to less accuracy in our food dataset because every picture is different from other with many colors and details. On the other hand, in our sign language dataset it consist of different hand shapes only so RGB is not necessarily required.. We also observe that if we train our model without shuffling datasets it was leading towards overfitting.

I also notice the effect of learning rates in our accuracy, different learning rate were tried but default was working best for us.

So we can say that our model worked decently giving high accuracy and low overfitting even with small dataset.