CMPT 412 – Assignment 5: Facial Recognition using CNN

By Prithi Pal Singh 301270075

CONTENTS

Objectives

The primary objectives of the assignment is to design and write a facial recognition classifier using Convolutional Neural Networks using MATLAB's neural net library. The whole framework is divided into three major parts: Data acquisition, setting up of layers and determination of accuracy of the neural net used. We already had the ATT face database from the previous assignment so this posed no challenge. The most time consuming part was setting up and choosing appropriate values of training neural net layers.

Neural Nets. Understanding

The firm grasp of CNN Layeral workings, the types of layers, their intended goals, parameters to tweak and their influence on performance as a whole were mandatory to understand before writing the implementation for our ATT Database set. Just Learning about the layers and their corresponding parameters are not enough to design one. So [1] provided a good layout on the importance of the parameters and even proposed a relational

$$\boldsymbol{O = (\frac{w-f+2p}{s}) + 1}$$

establishing connection with variables like Padding, Stride, Output dim and kernel dim.

My approach for layer setup and hyper-parameter determination is to go from simplest to a complex model. By Simple I mean number of layers, hyper-parameters such that they take small time to estimate the accuracy.

[1]. https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Legends for Conv Layer hyper parameters

W = input width

H = input height

Ow = output width

OH = output height

F = Kernel's Size

K = Number of Filters (Conv's depth)

S = Stride

P = zero-padding rows

Decisions

Choice of Conv. Hyperparams : These parameters dictates the computations required to process an image and information about neurons : their number, their local areas and convolution filters. The motivation to begin with this step is because I want perfect sized image dimensions to pass through each layer. For instance in Conv Layer, the relation $\boldsymbol{O} = (\frac{\boldsymbol{w-f+2p}}{\boldsymbol{s}})$ **+ 1**

helped me determine what values of Stride and Padding is required to keep my output dimension same as input from Convolution2dLayer.

Choice of Pooling technique: Max or average Pooling. That can be manipulated by two parameters Stride and Frame size. The relation of these two variables and the input, outputs dimensions are as follows [1]: $o = \left(\frac{w-f}{s}\right) + 1$ where o = output dim, w = input dim, f = frameSize and s = Strind.

Choice of data-reduction params: In CNN layers, these are often achieved by the pooling layer where appropriate FrameSize and Stride value can compress the data to the nth fraction of the input image * no. of filters.

Choice of Pre-processing: The pre-processing strategy may help in feature reduction. Techniques such as smoothing, different morphology such as opening and erosion are known to get rid of details and can be exploited to use.

Choice of Number of Filters:

Choice of filter numbers for each convolution: There are two approaches, either use same number of filters or use specific tested filter number for each convolution layer. For latter, there is no explanation for using those different explicitly values so If I have used them, I have conviction in their presence because their values helped me achieve better results.

Approach 1 : Using only one Conv Layer

Params :

To see how does things work around, I equipped CNN with just one Conv2dLayer. To decide upon the parameters,

```
layers = [
imageInputLayer([112 92 1])

convolution2dLayer(5,10,'Stride',1,'Padding',2)
reluLayer
averagePooling2dLayer([112 92])


fullyConnectedLayer(10)
softmaxLayer
classificationLayer

]
```

Here the accuracy comes around the ~ 0.5.

Approach 2 : Adding few more layers and after two layer reduce size to half. To half during pooling, the appropriate params are selected.

```
layers = [
imageInputLayer([112 92 1])  ;

convolution2dLayer(5,6,'Stride',1* ,'Padding',3);
reluLayer;

maxPooling2dLayer([2 2]);

convolution2dLayer(5,6,'Stride',1,'Padding',3);
reluLayer;

maxPooling2dLayer([2 2],'Stride',2);
```

```
convolution2dLayer(5,6,'Stride',1,'Padding',3);
reluLayer;

maxPooling2dLayer([2 2],'Stride',2); ## this has the dimension 28 * 23

fullyConnectedLayer(10);
softmaxLayer;
classificationLayer;
]
]
```

```
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|            1 |            1 |         6.51 |       3.6881 |        1.56% |       1.0000 |
|            3 |            3 |        18.64 |      13.8250 |       13.28% |       1.0000 |
```

Accuracy : 0.4 (with max/average pooling)

Observations :

- Here I added few more layers of Conv + RELU layers with two pooling in between. As you can see, both pooling layers are reducing data-size by half. So in the end right before the full-connected layer, total number of data = (112 / 2) /2 = 28. I believe having this number close to our final numCategories(10 here).
- In CNN layers, If we fix the Stride at 1, then the equation becomes independent of the input-size.
- The Depth of CNN layer (*NumFilters*) strongly increases the computational time because *NumFilters* computations would be required for each image now.

At average my accuracy was coming around no better than the 0.4 and many a just zeros. I noticed that learningRate should be kept minimum (0.001) or not greater than one for accuracy to be non-zero.

I experimented with different pre-processing strategies and this is how it went :

Laplacian of Gaussian

```
I = imread(filename);
x = fspecial('log',hsize,sigma);
log_im = imfilter(I,x);
FINAL=log_im;
```

Observations

NumFilters	HSize	sigma	Accuracy
10	10	25	0.25
10	20	25	0.25
10	2	5	0.25
20	2	5	0.25

These were the results provided that I have not tinkered the trainingOptions yet. After researching and reading the online documentation for MATLAB, I come to realize that I was using very low Epoches (3) and learningRate of 1. So I reset them to MaxEpoches = 30 and learningRate to 0.003. However the results were not quite impressive despite.

Approach 3 : Going back to one Conv layer and no pre-processing.

Next thing I did was to decrease the number of convolutional Layers all together and the pre-processing now. I removed the pre-processing because I wanted to achieve the optimal maximum accuracy without the pre-processing and then it would add the benefit on top of then layer configurations.

```
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|            1 |            1 |         1.28 |       4.8096 |        4.69% |       0.0010 |
|           10 |           10 |        11.01 |       0.8039 |       85.16% |       0.0010 |
|=========================================================================================|
```

where test accuracy was 0.5500. Quite nice this time. Now I increased the NumFilters to 20 from 10 and result was 0.55 . With no preprocessing and configuration below, I collected some numbers on different NumFilters.

```
layers = [
imageInputLayer([112 92 1])  ;

convolution2dLayer(5,10,'Stride',1,'Padding',3);
reluLayer;


maxPooling2dLayer([2 2],'Stride',2);
```

```
fullyConnectedLayer(40);
softmaxLayer;
classificationLayer;
]

options = trainingOptions('sgdm',...
    'InitialLearnRate',0.001,...
    'MaxEpochs',10, ...
    'Verbose',true,...
    'ExecutionEnvironment','auto');
```

NumFilters	Accuracy	Notes
10	0.5, 0.6, 0.49, 0.2	
20	0.52, 0.45	
30	0.47, 0.54	
35	0.71, 0.67, 0.54, 0.69	After this NumFilter, acc. Stays same.
40	0.63, 0.57	
60	0.74	
80	0.62	

Here, as we surpass the value 35, we tends to get either same or below accuracy. So let's stick to the 35 number of Filters.

Approach 4: Single Conv layer and morphology opening | Binary

Now, to test whether pre-processing poses any improvement, I used opening morphology operation and tweaked the value of disk of convolution. This is how it looked at disk size=3

Opening

With pre-processing at NumFields = 35, the resuts were 65 percent accurate and at 10, 10%. The thing to note is that It is hard to fix another value for Stride other than one because then any of the other ConvLayer parameters turns out to be fraction. So I am sticking to Stride to the 1 and calculating other following values using $\boldsymbol{O} = (\frac{\boldsymbol{w-f+2p}}{\boldsymbol{s}})$ **+ 1**

Again at max, I am able to reach ~ 0.7 of the accuracy with one convolution layer. If I add one more layer, It would take time beyond the capabilities of my system where MATLAB does not support GPU multi-clustering processes.

To tinker with the Pooling variables, I come at conclusion that $w = \frac{n}{n-1}\ with\ s = 1\ ; f = 2\ where\ n = fraction\ w.r.t\ o$. That means original-size / n = new-size.

Binary

Binarization gives out very pathetic results less than 0.01 of accuracy. Probably it loses on important features to help with identification.

Conclusion

The best approach I confronted was to setup layers and their values should be such that **S** is 1 and **O = Ow** and **H = OH.** This leaves to equation, KernelSize = 2*Padding size. Just try changes these values and select any appropriate NumKernels because it is independent in the equation and does not influences other variables.

Overall This is the best results I have got so far.

```
clc;
clear;
close all;
tic

imds =
imageDatastore('./face_database','IncludeSubfolders',1,'LabelSource','foldernames');
imds.ReadFcn = @(filename)morph_open_image(filename); ## opening with disk size=1


T = countEachLabel(imds)

trainNumFiles = 5;
[training_data,validation_data] =
```

```
splitEachLabel(imds,trainNumFiles,'randomize');
%%
layers = [

imageInputLayer([112 92 1])  ;

convolution2dLayer(3,35,'Stride',1,'Padding',1);
reluLayer;

maxPooling2dLayer([2 2],'Stride',2);


fullyConnectedLayer(40);
softmaxLayer;
classificationLayer;
]


%%

options = trainingOptions('sgdm',...
    'InitialLearnRate',0.001,...
    'MaxEpochs',40, ...
    'ExecutionEnvironment','parallel');
%%
net = trainNetwork(training_data,layers,options);


%%
predictedLabels = classify(net,validation_data);
valLabels = validation_data.Labels;
accuracy = sum(predictedLabels == valLabels)/numel(valLabels)

toc
```

This gives me,

```
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|           1 |           1 |         4.13 |       7.4372 |        3.12% |       0.0010 |
|          40 |          40 |       232.03 |       0.0000 |      100.00% |       0.0010 |
|=========================================================================================|
```

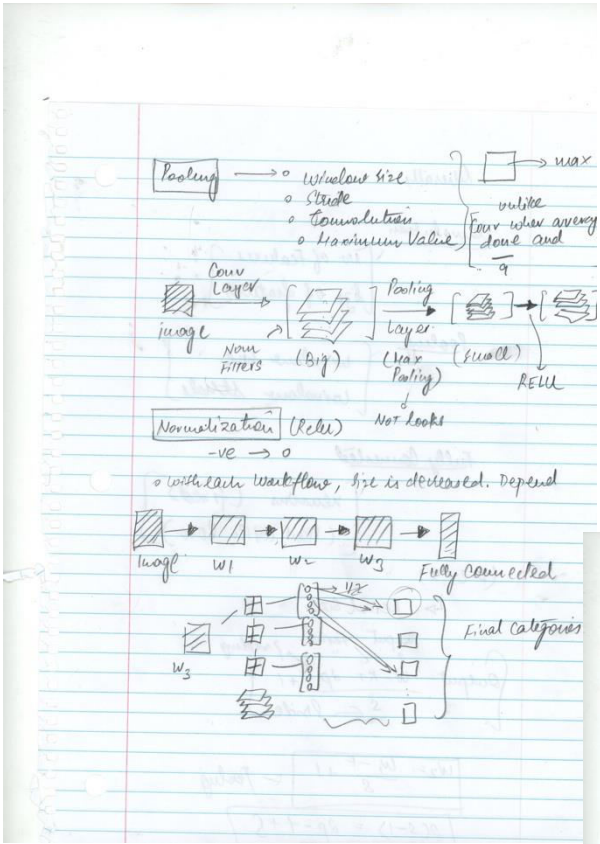
and

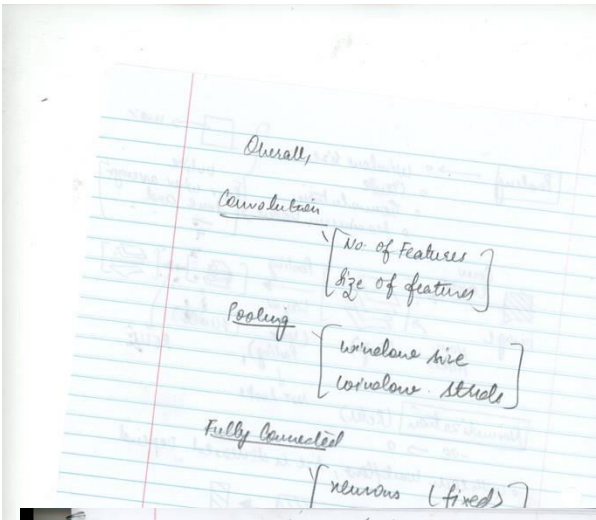
```
accuracy =


    0.7800
```

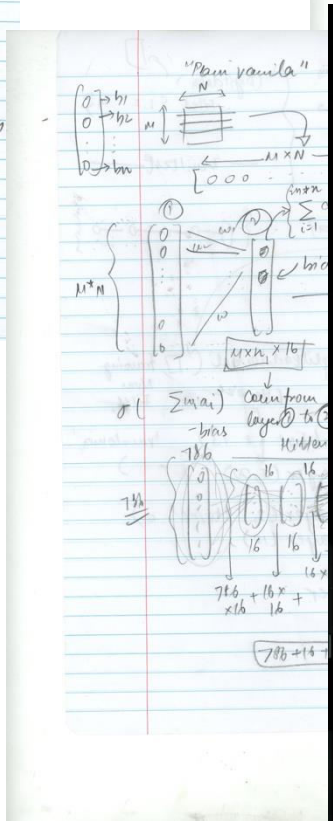
External Function :

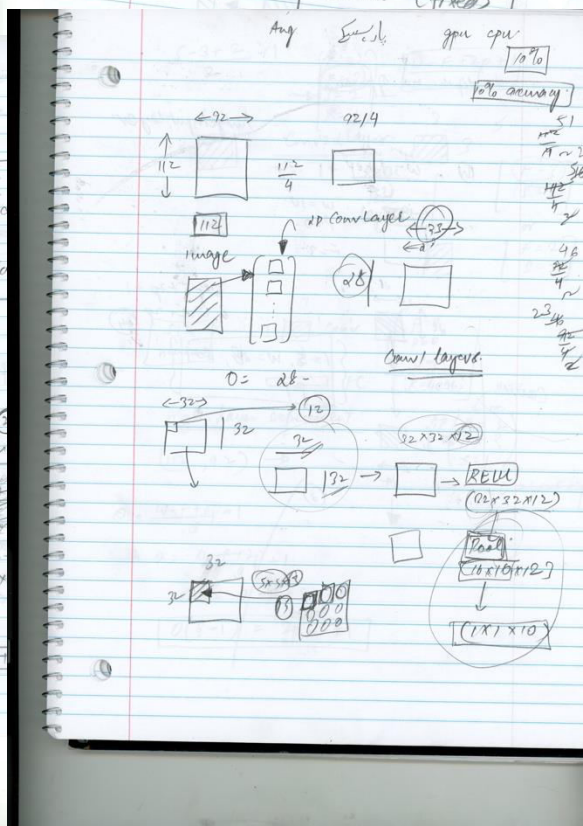
```
function FINAL=morph_open_image(filename)


n = 3;
I = imread(filename);
s = strel('disk',n);
morph_image = imopen(I,s);
FINAL=morph_image;


end
```

Rough Work









[filter, number of filters]

[redacted]

Convolution layer parameters

- Padding [a b] [t b l r]
- Filter Size [h w]
- NumFilters ~ output featuremaps
- Stride Step-size
- PaddingSize [1 1 2 2 / x x x x]
- Weights FilterSize(1) × FilterSize(2) × NumChannels × NumFilters
- Bias
- NumChannels

paddingsize


Examples
(11, 96, 'Stride', 4) — Padding normal 'manual' (0 0 0 0

11 → Filter size; 96 → Num Filters

- BiasLearnRateFactor

Overall,

Convolution
[No. of Features
Size of features]

Pooling
[window size
window stride]

Fully Connected
[neurons (fixed)
(Start with low)]

(I c=2)
↳ Kernel Size

$$\text{Output} = \frac{W - K + 2P}{S} + 1$$

(W: input, K: kernel, P: padding, S: stride)

$$W_2 = \frac{W_1 - F}{S} + 1 \leftarrow \text{Pooling}$$

$$O(S-1) = 2p - f + S$$

$2+1$

$$SW = W - f + 2p + S$$

$$W(S-1) = 2p - f + S$$ [40% maximum]

if $S = 1$

$$0 = 2p - f \therefore 2p = f$$

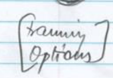
[116, 22, 1]

0.01

1.56% 4.69% 0%

fspecial('laplacian', alpha)
fspecial('log', hsize, sigma)

log_imag



Observations

NumF	hsize	σ	Acc.
10	10	25	0.25
10	20	25	0.25
10	2	5	0.250
20	2	5	0.250

↓ [illegible]