CENG 483

Introduction to Computer Vision

Spring 2018-2019

Take Home Exam 3

Image Colorization

Student Random ID: 13

1 Baseline Architecture (30 pts)

Max number of epochs is 50 in all the experiments at this part.

Based on your qualitative results (do not forget to give them),

- Discuss effect of the number of conv layers:

# of conv layers	kernel size	# of kernels	learning rate	accuracy
1	3	3	0.1	0.74
2	3	8	0.1	0.75
2	5	4	0.1	0.75
4	5	4	0.1	0.70
2	5	8	0.1	0.76
4	5	8	0.1	0.75

Table 1: # of conv layer

If we look at the table1 there is no significant difference when we change the number of conv layers. If the images in the dataset are different from each other in terms of perspective, color, shape etc. it would be clever to use more than 2 convolution layers because every conv layer will correspond to a different feature of the dataset. For example lets say we are training our model with horse images. In the network one layer will handle the horse figure which was taken in front of the horse , one will handle the back, up ,right or left of the horse. So it would make sense to have a bunch of conv layers in the network. But our dataset has images as human faces and there are not much perspective changes or color changes in the training images. So it is normal to have small changes in the accuracy when we change the conv layer number.

- Discuss effect of the kernel size(except the last conv layer):

  Size of the kernels play an important role to find the key features. Sometimes a larger size of kernel may skip the essential details in the image and it can lead you to underfitting. A smaller kernel can give more detailed information and it can lead you to overfitting.We need to determine the most suitable kernel size. In this experiment we have tried size of the kernel 3 and 5. According to table2 , when the kernel size increases from 5 to 3 ,generally the accuracy of the model increases. But there are no accurate results. This may be caused by the images in dataset.When the kernel size is

# of conv layers	kernel size	# of kernels	learning rate	accuracy
2	5	2	0.001	0.61
2	3	2	0.001	0.65
2	5	8	0.001	0.63
2	3	8	0.001	0.66
2	5	8	0.1	0.76
2	3	8	0.1	0.75
4	5	4	0.1	0.70
4	3	4	0.1	0.65
1	5	3	0.1	0.73
1	3	3	0.1	0.74

Table 2: kernel size

3 , it would extract more information from an image and this may increase the accuracy.Also the image sizes in the dataset are $80x80$ (it is not much) , so kernel size of 3 will give more detailed information rather than kernel size of 5.

- Discuss effect of the number of kernels(except the last conv layer):

# of conv layers	kernel size	# of kernels	learning rate	accuracy
2	5	2	0.001	0.61
2	5	8	0.001	0.63
2	3	2	0.001	0.61
2	3	8	0.001	0.66
2	5	8	0.1	0.76
2	5	4	0.1	0.75
4	5	4	0.1	0.70
4	5	8	0.1	0.75

Table 3: # of kernels

The purpose of a kernel in a CNN is to detect distinguishing features in the images. When the number of kernels increases in the network , one can extract more information from an image. For example using only one kernel for the images gives us a specific feature like round shapes , but it cannot generalize on let's say blue cars at the same time.Therefore convolution layers have several kernels to detect more/different features.Also too few kernels would possibly lose information and overfit to specific patterns, while too many kernels may underfit. Again we need to choose the most suitable kernel numbers.
When we look at the table3 ,it is easy to say that when the number of kernels is 8 (rather than 2 or 4) the model gives better results since there is more distinguishing features captured from kernels.Therefore learning is better.

- Discuss effect of the learning rate by choosing three values: a very large one, a very small one and a value of your choice:

  Learning rate is a hyper-parameter which controls how much we are adjusting the weights of our CNN with respect to the loss gradient. Also for the time efficiency of our model, choosing the learning rate is difficult task because a value too small may result in a long training process that

# of conv layers	kernel size	# of kernels	learning rate	accuracy
2	3	2	0.001	0.65
2	3	2	0.0001	0.23
2	5	8	0.001	0.63
2	5	8	0.1	0.76
2	3	8	0.001	0.66
2	3	8	0.1	0.75

Table 4: learning rate

could get stuck, while a value too large may result in learning an unstable training process. According to table4, one can see that when learning rate is greater , the accuracy of the model is better. Since there is a restriction on max number of epochs, smaller learning rates slow down the training process and will not allow the model learn enough. On the other hand,larger learning rates (0.1 here) gives better results.

2 Further Experiments (20 pts)

Max number of epochs is 50 in all the experiments at this part.

Based on your qualitative results (do not forget to give them),

- Try adding a batch-norm layer (torch.nn.BatchNorm2d) into each convolutional layer. How does it affect the results, and, why? Keep it if it is beneficial.
  Batch normalization normalizes the output of a previous activation layer by subtracting the batch

is BN added	# of conv layers	kernel size	# of kernels	learning rate	accuracy
no	2	5	8	0.1	0.76
yes	2	5	8	0.1	0.78
no	2	3	8	0.1	0.75
yes	2	3	8	0.1	0.76

Table 5: batch normalization

  mean and dividing by the batch standard deviation.It adds two trainable parameters to each layer, so the normalized output is multiplied by a “standard deviation” parameter and add a mean parameter. In other words, batch normalization lets stochastic gradient descent do the denormalization by changing only these two weights for each activation, rather than losing the stability of the network by changing all the weights.
  When we add batch normalization into each conv layer, there is an increase in accuracy (but not that significant) , it may be because of the images in dataset.(In table5)

- Try adding a tanh activation function after the very last convolutional layer. How does it affect the results, and, why? Keep it if it is beneficial.

  Activation functions determine the output of a deep learning model, its accuracy, and also the computational efficiency of training a model—which can make or break a large scale neural network.Putting tanh activation function after the very last convolutional layer,we want the output

is tanh added	# of conv layers	kernel size	# of kernels	learning rate	accuracy
no	2	5	8	0.1	0.76
yes	2	5	8	0.1	0.74
no	2	3	8	0.1	0.75
yes	2	3	8	0.1	0.72
no	2	5	4	0.1	0.75
yes	2	5	4	0.1	0.74

Table 6: tanh

data in format of +1s and -1s.
There is no significant difference when tanh is added or not. When there is no tanh in the network, output format is between [-1,+1] and tanh also makes the output format [-1,+1] so tanh may not affect the network much.

- Try setting the number of channels parameter to 16. How does it affect the results, and, why? Keep it if it is beneficial.
  According to table7, when we set the number of channels parameter to 16, there is no significant

rows	# of conv layers	kernel size	# of kernels	learning rate	accuracy
a1	2	5	4	0.1	0.75
	2	5	8	0.1	0.76
	2	5	16	0.1	0.76
a2	2	3	8	0.1	0.75
	2	3	16	0.1	0.75
a3	2	5	2	0.001	0.61
	2	5	8	0.001	0.63
	2	5	16	0.001	0.67

Table 7: # of channels param 16

change in the rows a1 and a2. As we discussed in $part1$ of this homework, kernels detect interest points in the images and this need to be tuned according to image size, accuracy of the experiments etc. Because dataset images sizes' are $80x80$ (they are not high resolution images) ,and so, there are some features that kernels need to detect but these features are not too many. In the row a2 and a1, 8 kernels and 16 kernels give the same accuracy that means they would approximately detect similar features. (increasing number of kernel 8 to 16 does not change the accuracy a lot).
In the row a3, accuracy of the network gets greater (but not much) , explanation of this may be the learning rate is not large enough so the model is not as good as other models.At this point increasing number of kernels may result in a higher accuracy by extracting more features than number of kernels 2 and 4 do.

3 Your Best Configuration (20 pts)

Using the best model that you obtain, report the following:

- The automatically chosen number of epochs(what was your strategy?):
  In every 5 epochs, the model calculates the validation loss and assigns it to the values named

loss1/loss2/loss3.
At the beginning loss1 holds the first validation loss. After 5 epochs loss2 holds the second validation loss and after 10 epochs, loss3 holds the third one. When loss1 $<$ loss2 $<$ loss3 , the networks stops. (should_stop function controls this condition). In other words , when validation loss starts to increase , it stops.

- The plot of the training mean-squared error loss over epochs:
  Training mean-squared error loss has decreased significantly at the first 15 epochs.(approximately).After that it barely changes, also it is a good thing that it never increases so much.

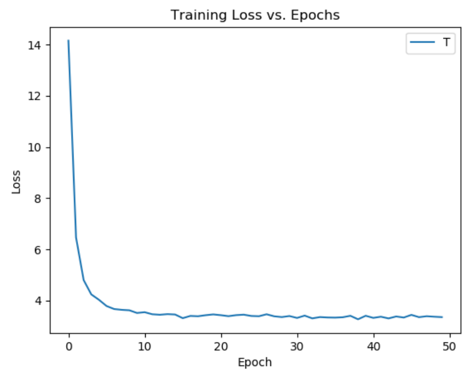

Figure 1: Training mean-squared error loss over epochs

- The plot of the validation 12-margin error over epochs (see the3 text for details):
  It computes the ratio of correctly estimated pixels to all of them , correct means having estimated with an error margin of 12.The graphic has peaks at every 25 epochs.

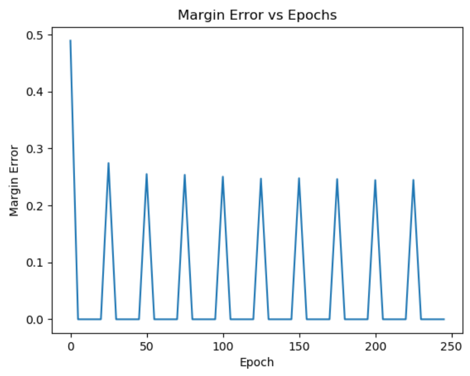

Figure 2: 12-margin error over epochs

- At least 5 qualitative results on the validation set, showing the prediction and the target colored image:
  I have choosed the images whose accuracy is greater than 0.70.
  Figure 3- Figure 4-Figure 5-Figure 6-Figure 7 shows these images.

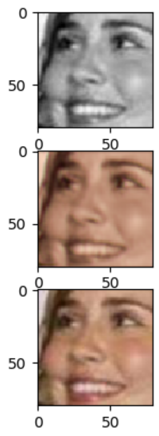

Figure 3:

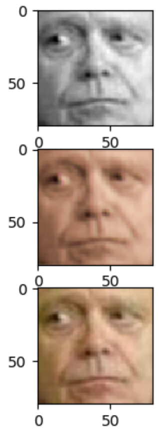

Figure 4:

- Discuss the advantages and disadvantages of the model, based on your qualitative results, and, briefly discuss potential ways to improve the model:

  One disadvantage of my model is the stop condition. When repetitive three training losses(named loss1, loss2 , loss3 here) start to increase, the model stops training(when loss1 ¡ loss2 ¡ loss3) , that is a good stop condition but it does not save the loss1's weights ie. the model stops when loss3 appears and loses loss1's weights. In order to improve this, we can hold the weights when loss1 appears.

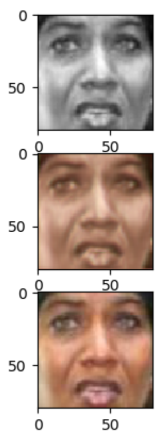

Figure 5:

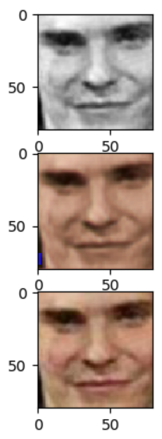

Figure 6:

Another disadvantage is the max number of epochs.The max number of epochs is 50. If the code does not be captured in the stop condition it should train the model more than 50 epochs to get better results. For time efficiency, I have choosed the max number of epochs 50 but I think it should be at least 100.
Advantages of my model is there is a batch normalization after each layer and according to experiments in part1, I have choosed best hyper-parameters.(number of conv layer = 2,kernel size = 3 , number of kernels 8 , learning rate 0.1 , batch size = 16)

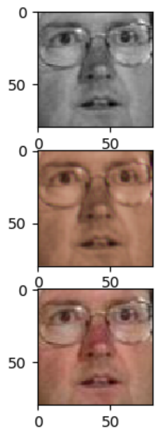

Figure 7:

4 Your Results on the Test Set(30 pts)

This part will be obtained by us using the estimations you will provide. Please tell us how should we run your code in case of a problem:
This is my CNN class for my best configuration.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(1, 8, 3, stride=1, padding=1)
        self.conv1_bn = nn.BatchNorm2d(8)

        self.conv2 = nn.Conv2d(8, 3, 3, stride=1, padding=1)
        self.conv2_bn = nn.BatchNorm2d(3)


    def forward(self, grayscale_image):

        x = F.relu(self.conv1_bn( self.conv1(grayscale_image) ))
        x = self.conv2_bn(self.conv2(x) )
        return x
```

There is a function named should_stop which controls the stop condition.

```
def should_stop(loss1,loss2,loss3):
    if(loss1< loss2 < loss3):
        return 1
    else:
        return 0
```

This part automatically choose the number of epochs, by testing the validation-set performance every 5 epochs. First assign loss1 , loss2 and loss3 to zero which we use in stop condition.Then create an empty list named estimations. We will save the validaition predictions into this in every 5 epoch . Here we change the dimensions of prediction to (80, 80, 3) and values in the array in the range [0,255].
Then we shift the loss values and check whether it sould stop or not.

```
print('training begins')
loss1=0
loss2=0
loss3=0
estimations=[]
for epoch in range(max_num_epoch):
    running_loss = 0.0 # training loss of the network

    for iteri, data in enumerate(train_loader, 0):
        inputs, targets = data # inputs: low-resolution images, targets: high-↩
           resolution images.

        optimizer.zero_grad() # zero the parameter gradients

        # do forward, backward, SGD step
        preds = net(inputs)

        loss = criterion(preds, targets)
        loss.backward()
        optimizer.step()

        # print loss
        running_loss += loss.item()

        print_n = 100 # feel free to change this constant
        if iteri % print_n == (print_n-1):    # print every print_n mini-batches
            print('[%d, %5d] network-loss: %.3f' %
                  (epoch + 1, iteri + 1, running_loss / 100))


            running_loss = 0.0

        if (iteri==0) and VISUALIZE:
            hw3utils.visualize_batch(inputs,preds,targets)

    net.eval()
    if( epoch % 5 == 0  ):
        estimations=[]
        vl=0.0
        for i,data2 in enumerate(val_loader,0):

            val_inputs, val_targets =  data2

            val_preds = net(val_inputs)
            for i in range(0,16):
                x = val_preds[i]

                y = x.detach().numpy()

                z= np.transpose(y , (1,2,0))
                z= (z+1) * 127.5

                estimations.append(z)


            val_los = criterion(val_preds, val_targets)
            vl+=val_los.item()

        if(loss1==0):
            loss1=vl
        elif(loss2==0):
            loss2=vl
        else:
            loss1=loss2 #her bir val loss kontrolunde loss'lar bir kayiyor
            loss2=loss3
            loss3=vl
    net.train()
    if(should_stop(loss1,loss2,loss3)==1):
        break

    print('Saving the model, end of epoch %d' % (epoch+1))
```

We change the "estimations" list to a numpy array and save in a file named *estimations_validation.npy*:

```
estimations = np.asarray(estimations)
np.save('estimations_validation',estimations)
```

At the end,we compute the test images:

```
for i,data3 in enumerate(test_loader,0):

            test_inputs, test_targets =  data3

            test_preds = net(test_inputs)
            for i in range(0,16):
                x = test_preds[i]
                y = x.detach().numpy()

                z= np.transpose(y , (1,2,0))
                z= (z+1) * 127.5

                estimations_test.append(z)



estimations_test = np.asarray(estimations_test)
np.save('estimations_test',estimations_test)
```

5 Additional Comments and References

https : //towardsdatascience.com/batch − normalization − in − neural − networks − 1*ac*91516821*c*
https : //pytorch.org/tutorials/beginner/blitz/cifar10_t*utorial.html*
https : //blog.algorithmia.com/convolutional − neural − nets − in − pytorch/
*http : //user.ceng.metu.edu.tr/ gcinbis/courses/Spring*19*/CENG*483*/*
This homework helps a lot to figure out how CNN works, thank you.