CENG 483

Introduction to Computer Vision

Spring 2018-2019

Take Home Exam 3

Image Colorization

Student Random ID: 98

1 Baseline Architecture

The number of convolutional layers tested: 1, 2, 4

- Considering the fact in the item 1 of section 5, it is clear that models with more than one layer will always have much greater parameter capacity than single-layered models due to the opportunity of having a varied number of kernels in the intermediate layers in addition to the extra levels of layers. Generally, we can expect a better performance from the models with greater capacity. However, when the capacity is increased beyond the ideal, it might take a longer time to converge and, after some point, it is expected that excessive capacity will result in overfitting and worse performance.
- The following configuration was run for 10 epochs:
  - Batch size: 16
  - **Kernel size: 3**
  - Kernel count: 2
  - Learning rate: 0.1
  - **CONV layers: 1, 2, 4**
- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
1-layered	0.759	0.010
2-layered	0.764	0.010
4-layered	0.742	0.011

- The following configuration was run for 10 epochs:
  - Batch size: 16
  - **Kernel size: 5**

  - Kernel count: 2
  - Learning rate: 0.1
  - **CONV layers: 1, 2, 4**

- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
1-layered	0.760	0.010
2-layered	0.762	0.010
4-layered	0.673	0.021

- Based on the experimental results, it can be concluded that among the suggested values the ideal capacity is achieved when there are 2 layers (with possibly larger kernel counts as other experiments not shown here suggest). They indicate that 2-layered model is good enough to achieve adequate complexity with fast convergence and accurate results. Similar results are achieved under different hyperparameter settings.

The kernel sizes tested: 3, 5

- The following configuration was run for 10 epochs:
  - Batch size: 16
  - **Kernel size: 3, 5**
  - **Kernel count: 4**
  - Learning rate: 0.1
  - CONV layers: 2

- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
Kernel Size: 3	0.772	0.010
Kernel Size: 5	0.766	0.009

- The following configuration was run for 10 epochs:
  - Batch size: 16
  - **Kernel size: 3, 5**
  - **Kernel count: 8**
  - Learning rate: 0.1
  - CONV layers: 2

- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
Kernel Size: 3	0.774	0.009
Kernel Size: 5	0.774	0.009

- Based on these and other experiments not shown here kernel sizes 3 and 5 do not seem to have a superiority over each other in terms of the loss and accuracy rates. Depending on the hyperparameter settings one can be slightly better than the other. Since no significant effect on the performance is noticed, in practise, both can be used.
- All images in the data set only include faces and they are all centered. In other words, there is no significant variance among different pictures in the data set. Moreover, they are all of size 80x80x3. In this experiment, increasing the size of kernel form 3 to 5 only adds 16 pixels to the receptive field of each hidden unit. Probably, a difference of 16 pixels do not make a significant difference in terms of features extracted and learned from this specific data set.

The number of kernels/channels tested: 2, 4, 8

- It is expected that increasing number of kernels improves performance because of its contribution to model capacity.
- Considering the fact in the section 5 item 1, single convolutional layer is not used in the tests for the number of kernels because the number of kernels will be omitted.
- The following configuration was run for 10 epochs:
  - Batch size: 16
  - Kernel size: 5
  - **Kernel count: 2, 4, 8**
  - Learning rate: 0.1
  - **CONV layers: 2**
- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
Kernel Count: 2	0.762	0.010
Kernel Count: 4	0.766	0.009
Kernel Count: 8	0.774	0.009

- The following configuration was run for 10 epochs:
  - Batch size: 16
  - Kernel size: 5
  - **Kernel count: 2, 4, 8**
  - Learning rate: 0.1
  - **CONV layers: 4**
- The maximum accuracy and minimum losses observed are:

	Max Accuracy	Min Loss
Kernel Count: 2	0.673	0.021
Kernel Count: 4	0.737	0.011
Kernel Count: 8	0.754	0.010

- Based on the experiments, it can be concluded that kernel count provides a more capable model due to the better capacity it offers. Experiments verify that it has a positive effect on the accuracy and loss, and thus on the performance. Therefore, the results meet the expectation.

Learning rates tested: 0.0001, 0.1, 10000

- Experiments show that the learning rate **0.0001** is too small to begin with. Suggested parameter combinations are tested up to 10 epochs while the learning rate is kept constant at this rate in all experiments. Results show that all models start with low accuracy as a result of poor coloring, which can be observed by visual inspection. Each model has a trend towards increasing the accuracy through epochs. However, a small number of epochs is not adequate to achieve the desired performance. After 10 epochs in 14 different experiments, the maximum accuracy achieved is only: 0.28 (at the end of 10th epoch of test with hyperparameters: batch size: 16, kernel size: 5, kernel count: 8, CONV layers: 2)

- The learning rate **0.1** provided results with higher accuracy in a smaller number epochs. Again all models are run until the end of 10th epoch and 14 different experiments performed. The best accuracy observed is: 0.77 (at the end of 9th epoch of test with hyperparameters: batch size: 16, kernel size: 5, kernel count: 8, CONV layers: 2)

- The learning rate **10000** produced NaN loss values due to overflow. A single experiment was performed to show that this learning rate is too large and not useful for training. Other hyperparameters than learning rate do not make such a large effect that can result in overflow in loss value. This fact indicates that such a huge learning rate disrupts training no matter how good other hyperparameter choices are. Therefore, no further combinations were tested with this learning rate.

2 Further Experiments

Adding Batch Norm:

- Configuration tested:
  - Batch size: 16
  - Kernel size: 5
  - Kernel count: 8
  - Learning rate: 0.1
  - CONV layers: 2
  - **Batch Norm: True, False**
  - tanh: False

- Using batch norm seems to provide slightly better accuracy and loss values. The model with batch norm starts with a lower validation loss and higher training loss. Maximum, minimum accuracy and loss values are slightly better with the batch norm.

	Max Accuracy	Min Accuracy	Max Loss	Min Loss
With Batch Norm	0.789 (E: 27)	0.721	0.012	0.008 (E: 31)
Without Batch Norm	0.783 (E: 17)	0.713	0.013	0.009 (E: 27)

Table 1: Statistics Related to Effect of Batch Norm: "E" indicates epoch number

- However, it takes more epochs to achieve these maximum accuracy and minimum loss values than the ordinary model. Overall, the difference is not huge but the model seems to have a more consistent accuracy plot.
- One reason why batch norm did not make a huge difference might be due to having inputs already zero-centered in the first place and the reason why it is slightly better than the ordinary model can be due to the regularization effect of batch norm. Since regularization is not explicitly done in the network, batch norm could have improved the performance slightly.

Adding tanh:

- Configuration tested:
  - Batch size: 16
  - Kernel size: 5
  - Kernel count: 8
  - Learning rate: 0.1
  - CONV layers: 2
  - Batch Norm: False
  - **tanh: True, False**

	Max Accuracy	Min Accuracy	Max Loss	Min Loss
With tanh	0.783 (E: 30)	0.692	0.013	0.009 (E: 36)
Without tanh	0.783 (E: 17)	0.713	0.013	0.009 (E: 27)

Table 2: Statistics Related to Effect of tanh: "E" indicates epoch number

- Adding tanh to the last layer does not seem to have an effect on the performance. In fact minimum accuracy value achieved is lower in this case. However, in some cases, it seems that final tanh layer was able to correct some wrong color choices in the images. Probably, these errors stem from numerical problems and final activation forces values to stay in [-1, 1] range, resolving possible overflow and underflow problems. Yet, such a situation occurred at 20th epoch, the model without tanh might have already corrected it later and it is also possible for tanh to perform worse in some cases.
- It is important to note that without tanh, model is able to reach to the same loss and accuracy value in less number of epochs.
- tanh is known to provide a better performance if the input is not zero centered. However, in this case, input is already zero-centered in the beginning (in the range: [-1, 1]) and even if the input was not centered, adding tanh to only the last layer would most probably not resolve this problem. Therefore, it is not surprising to see that tanh did not improve the performance.

Kernel Count: 16

- Configuration tested:
  - Batch size: 16
  - Kernel size: 5
  - **Kernel count: 8, 16**

- Learning rate: 0.1
- CONV layers: 2
- Batch Norm: False
- tanh: False

	Max Accuracy	Min Accuracy	Max Loss	Min Loss
Kernel Count: 16	0.797 (E: 32)	0.659	0.012	0.008 (E: 28)
Kernel Count: 8	0.783 (E: 17)	0.713	0.013	0.009 (E: 27)

Table 3: Statistics Related to Effect of Kernel Count: "E" indicates epoch number

- Increasing the number of kernels or (number of channels) seems to have a positive impact on maximum accuracy. However, loss ranges are almost the same and, in the loss and accuracy plots, some sharp fluctuations that are not present in the model with kernel count 8 are observed. The effect of such a fluctuation can be seen on the minimum accuracy value achieved with kernel count 16.
- Using more kernels increases the parameter capacity of the model. The increase in the capacity is likely to be the reason for the rise in the performance. However, results also show that the epochs required to obtain a higher performance increase too. Therefore, this parameter setting does not improve the performance in a considerable amount either.

3 Your Best Configuration

In the earlier section, it was observed that introducing different kinds of layers and increasing the number of kernels do not have a significant impact on the performance. Moreover, increasing the complexity results in longer training times. Overall, their contribution is not in an extent which is able to compensate the overhead of using them. Therefore, those settings are discarded and the following configuration is accepted as the best configuration:

- Batch size: 16
- Kernel size: 5
- Kernel count: 8
- Learning rate: 0.1
- CONV layers: 2
- Batch Norm: False
- tanh: False

Early Stop Heuristic: Since the main metric to evaluate the performance of the model is accuracy, the accuracy values obtained from the validation set is used for the stopping condition. In this approach, there are two triggers:

- The number of consecutive degradation in accuracy: Count the number of successive drops in the accuracy rate. If this number exceeds a "patience" value, then terminate training. Whenever a rise happened before the "patience" value, reset the counter.

- The number of consecutive epochs without a maximum accuracy: Hold maximum accuracy rate and check after each validation if a new maximum is achieved. Unless a new maximum rate is found after a certain number of epochs, terminate training. Whenever a new maximum achieved, reset the counter.

With this approach, training has stopped after 28th epoch. This training yield the following plots:

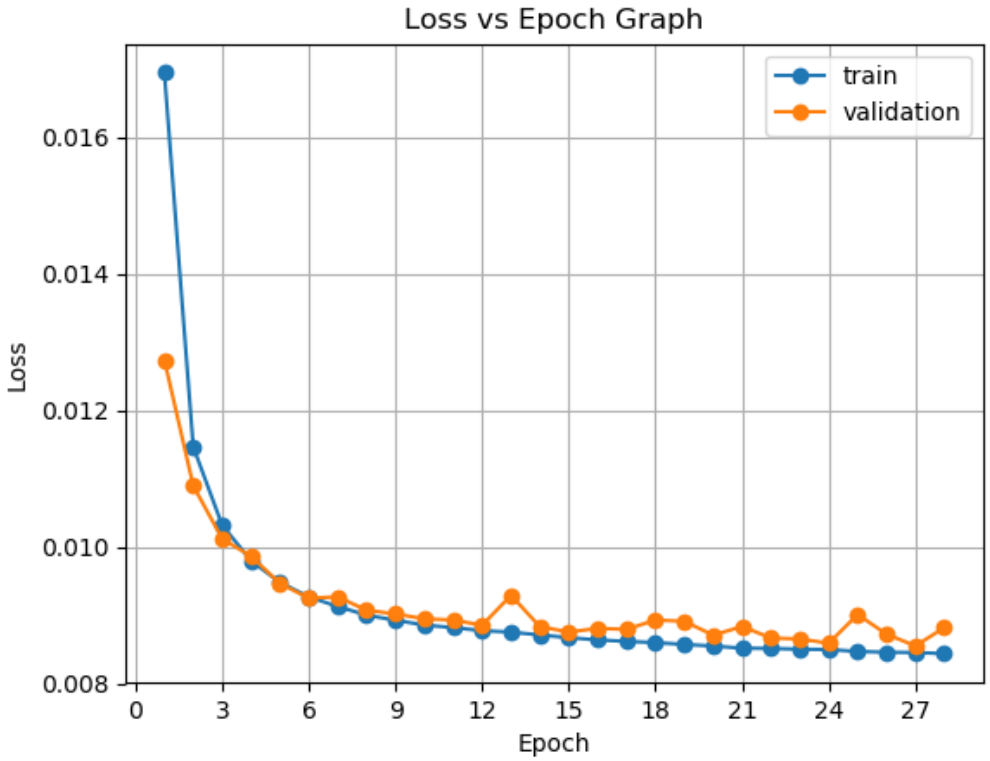

Figure 1: Accuracy vs Epoch Plot for the Best Configuration

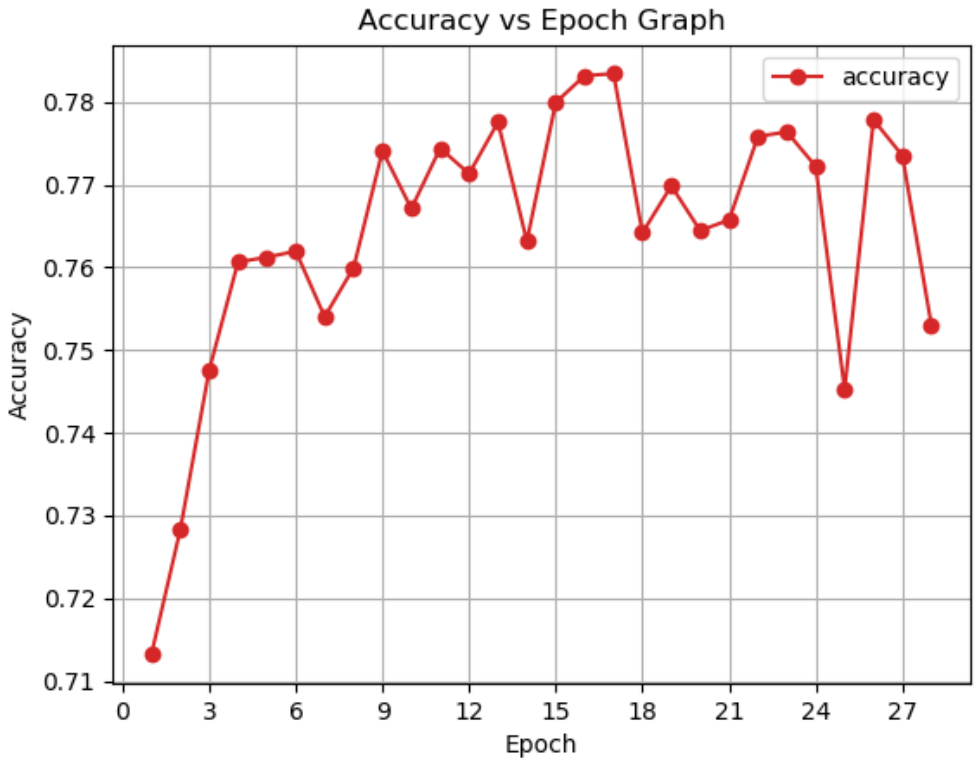

Figure 2: Accuracy vs Epoch Plot for the Best Configuration

Figure 3: Sample Coloring for Validation Set with the Best Configuration: Epoch: 17 with Accuracy: 0.78 & Validation Loss: 0.009

The model only uses 2 convolutional layers with 1 ReLU activation without any additional layers. In this sense, it is cheaper than most of the deeper and complex models. Despite the relatively low complexity, the model is able to converge to a high-accuracy, low-loss level much faster than many models. It only takes 17 epochs to reach 78% accuracy and 0.009 loss.

Regularization is not applied to the model. Therefore, the ability of model to make generalized predictions is probably low. The application of dropout or L2 regularization is likely to improve the performance. Most of the related works for image colorization include a pipeline consisting of a downsampling followed by an upsampling unit. It is possible to add pooling (maxout etc.) for downsampling and extra convolutional layers with large spatial sizes for upsampling.

4 Your Results on the Test Set

Using the "evaluate.py" script achieved accuracy is 0.74. During the testing, averaged accuracy rate among batches is calculated as 0.81.

Following command can be used to colorize test set images by loading the network from the checkpoint where highest accuracy was observed:

```
./the3.py -p test -bs 16 -ks 5 -kc 8 -lr 0.1 -cl 2
--gpu --checkpoint checkpoint_e17.pt
```

Note: It is assumed that test set path is as follows:

```
./ceng483-s19-hw3-dataset/test/images/
```

You can find full list of arguments by running the script as:

```
./the3.py -h
```

5 Additional Comments and References

1. Due to the suggested network architecture, when the number of convolutional layers is 1, the number of kernels does not have any effect on the performance. When there is only one layer, this parameter is omitted in order to set the number of kernels to be 3 to produce an RGB color image as the output from the single layer.

2. All accuracy and loss values of epochs mentioned in the report are **calculated by averaging the average values obtained from each batch with respect to all batches**. This method is used in order to speed up the accuracy calculations and to use less memory. As a result, "evaluate.py" script was not used within the validation. "evaluate.py" results might differ because of the way accuracy calculated. However, although the accuracy calculation from batches during the validation (used in this report) might be somewhat misleading, optimizing it also optimizes the actual accuracy calculation performed by "evaluate.py".

3. In all experiments, validation frequency is 1 epoch. In other words, after each epoch, validation and accuracy calculations are performed.

4. In all experiments, seed value used is 10.