# CENG 483

Introduction to Computer Vision

Fall 2021-2022

Take Home Exam 3

Image Colorization

Student ID:

## 1 Baseline Architecture (30 pts)

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

**The baseline hyperparameter configuration is as follows, and only the one discussed in the item is altered:**
**learning rate = 0.05, number of kernels = 8, kernel size = 5, number of convolutional layers = 2**

- Discuss effect of the number of conv layers:

  **Validation accuracy with number of conv layers = 1: 0.74**
  **Validation accuracy with number of conv layers = 2: 0.75**
  **Validation accuracy with number of conv layers = 4: 0.69**

  Adding layers to a neural network obviously increases the model complexity. And after a level of complexity the model becomes too specialized to the training set, hence, fails to do well on the validation set, ie, it overfits. Here, increasing the number of conv layers from one to two had a positive impact on the performance, however, this is not the case when it is increased to four. One probable reason for the drop in the accuracy is overfitting after two layered-model.

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

  **Validation accuracy with number of kernel size = 3: 0.76**

  Decreasing the kernel size from 5 to 3 while keeping all the other hyperparameters as baseline, had a positive effect on the model performance. Kernel size defines the window through which important parts of the image is searched. That is, with smaller kernel sizes, the window in which key/important points or defining objects/characteristics is smaller. The rise in performance might be caused by the small features in images are more informative. In particular, the color information is local and moving the window even by small amounts makes a difference. That's why there is an increase in the accuracy with smaller a kernel size.

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

  **Validation accuracy with number of kernels = 2: 0.75**
  **Validation accuracy with number of kernels = 4: 0.74**
  **Validation accuracy with number of kernels = 8: 0.75**

  The number of kernels kind of defines the number of different sets of information gathered from a window in the image. There is no constant decrease or increase in the model performance with more number of kernels. This is due to the fact that there is not many distinct sets of information to collect regarding color.

- Discuss effect of the learning rate by choosing three values: a very large one, a very small one and a value of your choice:
  **Validation accuracy with a small learning rate = 0.001: 0.69**
  **Validation accuracy with a large learning rate = 0.1: 0.73**
  **Validation accuracy with another learning rate = 0.05: 0.75**

  The learning rate defines the step of change in weights, ie, kernel parameters in this case, on a misclassification. Considering the search space where the objective function is optimized, learning rate defines how much to mode towards the gradient, ie, local minimum. However, large learning rate might cause to overstep minima, one/some of which might be the global minimum/a. On the other hand, small learning rate is known to converge more slowly, and it is possible to get stuck at a local minimum. Here we see the effect of learning rate as explained, small learning rate, ie, 0.001, performed rather poorly, most probably because it failed to converge being stucked at a local minimum. Large learning rate, although performed better than the smallest one, is still worse than the other. Large learning rate means a faster convergence, but what is converged to may not be optimal, especially when using an early stopping method working on the loss as I did here.

# 2 Further Experiments (20 pts)

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.

| num layers | kernel size | num kernels | learnig rate | batch norm | accuracy |
|---|---|---|---|---|---|
| 1 | 5 | 8 | 0.05 | no | 0.74 |
| 1 | 5 | 8 | 0.05 | yes | 0.72 |
| 2 | 5 | 8 | 0.05 | no | 0.75 |
| 2 | 5 | 8 | 0.05 | yes | 0.75 |
| 2 | 5 | 2 | 0.05 | no | 0.75 |
| 2 | 5 | 2 | 0.05 | yes | 0.74 |
| 2 | 5 | 4 | 0.05 | no | 0.74 |
| 2 | 5 | 4 | 0.05 | yes | 0.75 |
| 2 | 5 | 8 | 0.1 | no | 0.73 |
| 2 | 5 | 8 | 0.1 | yes | 0.76 |
| 2 | 5 | 8 | 0.001 | no | 0.69 |
| 2 | 5 | 8 | 0.001 | yes | 0.72 |
| 4 | 5 | 8 | 0.05 | no | 0.69 |
| 4 | 5 | 8 | 0.05 | yes | 0.72 |
| 2 | 3 | 8 | 0.001 | no | 0.76 |
| 2 | 3 | 8 | 0.001 | yes | 0.75 |

The table above shows the effect of batch normalization on different hyperparameter configurations when analyzed 2 lines at a time. Batch normalization normalizes the layers' inputs, so it is known to make the training process faster and provide better performance. Only significant change in the model performance observed is for the 4-layered network. And, batch normalization is known to better prevent overfitting, and this might be the reason for such an increase in the performance of the 4-layered more complex model, which is prone to overfitting.

- 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.

| num layers | kernel size | num kernels | learnig rate | tanh at the end | accuracy |
|---|---|---|---|---|---|
| 2 | 5 | 8 | 0.05 | no | 0.75 |
| 2 | 5 | 8 | 0.05 | yes | 0.74 |
| 2 | 3 | 8 | 0.05 | no | 0.76 |
| 2 | 3 | 8 | 0.05 | yes | 0.75 |
| 2 | 5 | 2 | 0.05 | no | 0.75 |
| 2 | 5 | 2 | 0.05 | yes | 0.73 |
| 2 | 5 | 8 | 0.1 | no | 0.73 |
| 2 | 5 | 8 | 0.1 | yes | 0.74 |

Adding tanh after the last layer yielded almost no improvement. The results are already in the same range as the output range of tanh, which is why it made almost no difference.

- Try setting the number of channels parameter to 16. How does it affect the results, and, why? Keep it if it is beneficial.

| num layers | kernel size | num kernels | learnig rate | accuracy |
|---|---|---|---|---|
| 2 | 5 | 16 | 0.05 | 0.76 |
| 2 | 5 | 8 | 0.05 | 0.75 |
| 2 | 3 | 16 | 0.05 | 0.76 |
| 2 | 3 | 8 | 0.05 | 0.76 |
| 2 | 5 | 16 | 0.1 | 0.73 |
| 2 | 5 | 8 | 0.1 | 0.76 |

The number of output channels of a convolutional layer is proportional to the number of kernels. Adding more kernels improved accuracy only slightly, that is as discussed above, ie, not much different sets of information to gather from a window in an image.

# 3 Your Best Configuration (20 pts)

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

Although there is no significant improvement on the model performance with the experimented hyperparameters, the model with the following hyperparameters is chosen: num layers = 2, kernel size = 5, num kernels = 16, learning rate = 0.05, which in the end yielded 0.76 accuracy for the validation set ,which is one of the highest achieved.

- The automatically chosen number of epochs(what was your strategy?):
  The networks perform better and better with more iterations of training on the training set, however, their learnable parameters, ie, weights become too specialized for the training set, so it fails to generalize well with the unseen data. That's why after the point where the validation loss starts to increase and keeps on increasing (may be not monotonously) is a good point to stop the training procedure to avoid overfitting. This is known as Vapnik's structural risk minimization.
  And the training is early stopped at epoch = 44 for this configuration.

- The plot of the training mean-squared error loss over epochs:

  The following figure shows the train and validation losses over epochs

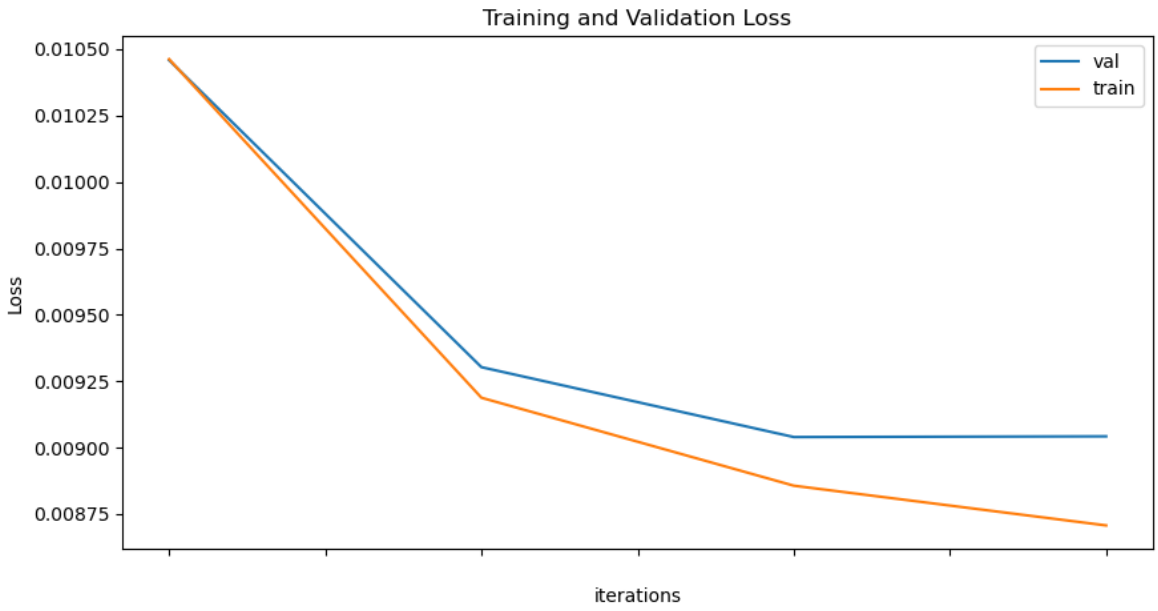


Figure 1: Train and validation losses over epochs

- The plot of the validation 12-margin error over epochs (see the3 text for details):
- At least 5 qualitative results on the validation set, showing the prediction and the target colored image:

  The following figure shows 5 of the colorized images form the validation set, where the ones on the first line are the input, the last line are the ground truth, and the ones in the middle are the predictions.

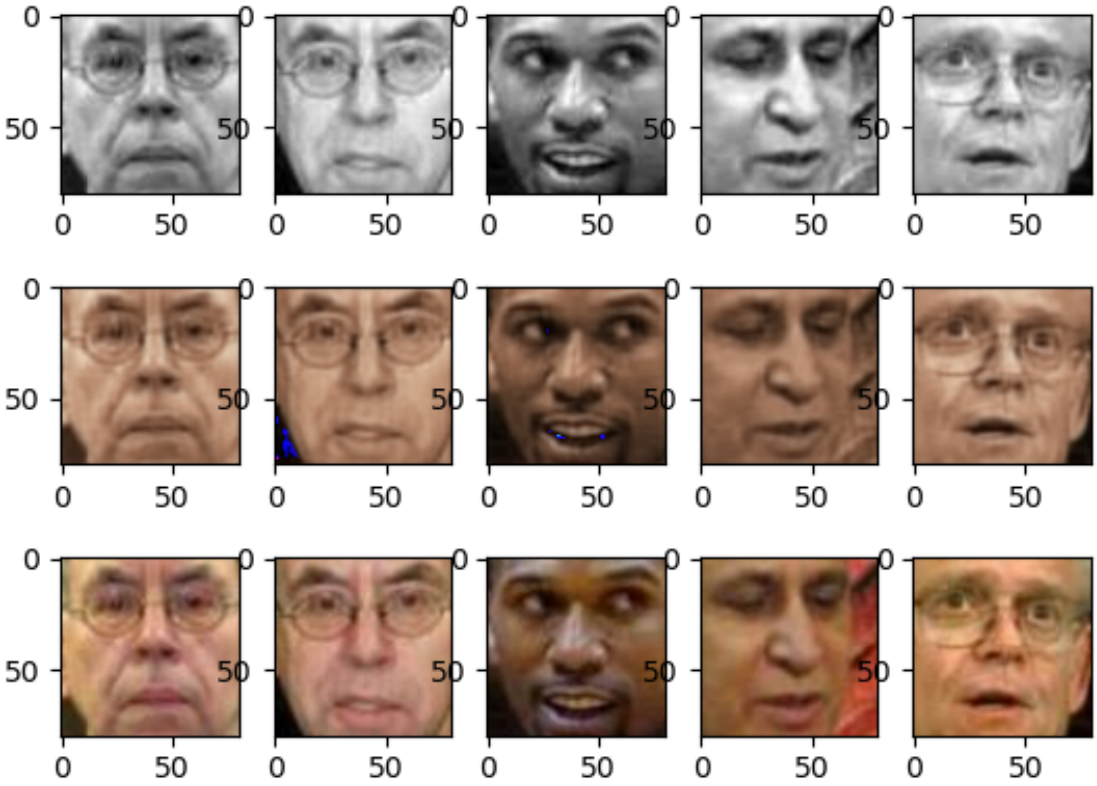

Figure 2: Colored images

- Discuss the advantages and disadvantages of the model, based on your qualitative results, and, briefly discuss potential ways to improve the model:
  The obtained accuracy and the colorized images are, although not perfect, acceptable. Other hyperparameter configurations could be used for tuning, ie, one could apply grid/random search. In fact, there are lots of configurations I did not try with the hyperparameter set I used. Also, there are other hyperparameters that we did not experiment on, such as dilation.

# 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:

When the.py is run with the default parameters set in the script, it first uses train and validation sets whose relative paths are "ceng483-s19-hw3-dataset/train/images" and "ceng483-s19-hw3-dataset/val/images" to train the model. Then, it automatically loads the test set images under "ceng483-s19-hw3-dataset/testinputs/ima and saves the first 100 predictions to "estimationstest.npy" and the corresponding images' paths to "testimages.txt" files, as stated in the homework text.