# CENG 483

Introduction to Computer Vision
Fall 2023-2024

Take Home Exam 3: Image Colorization

Full Name: Emirhan Yılmaz Güney
Student ID: 2443208

## 1 Baseline Architecture (30 pts)

As the first step, I have designed an experiment setup for a comprehensive analysis of the various parameters of an image colorization system, including the architecture and training loop details. For this purpose, I needed to both observe the quality of the training process, while training happens, and the quality of the output predictions the model has done, after the training has been completed. To achieve this, I used some different measurement metrics both visually and mathematically.

To analyse the quality of the training process in the training loop, I have added a validation loss which is calculated on the validation set provided each epoch. In addition, I have printed out both the training loss and validation loss values to visually observe what is going on in terms of losses.

After the training process, I have plotted the training and validation loss curves over epochs. These curves were beneficial in terms of observing the effect of learning rate especially, whether it is too large or too low.

In addition to the observations above, I have calculated the overall accuracy over the validation set to understand how well the model is doing and then I plotted and compared some of the predictions and the corresponding ground truths visually. The comparison was crucial to observe the abnormalities if there exist. For example, a high learning rate can cause some noises on the output, which decreases the accuracy.

To examine the effect of arhitectural parameters such as number of conv layers, number of kernels etc. I conducted controlled experiments, by letting all parameters stay constant except one of them, the one I am examining, and decide if model makes better or worse after this change by using the measurement metrics I have explained above. For example, an increase of the learning rate parameter may leave the accuracy and the last validation loss same, but this does not mean the performance is same before checking the predictions visually. An example to the noisy output is shown in Figure 1. The reason of this noise may be the wrong learning rate choose, the wrong stopping criteria etc.

### 1.1 Effect of Number of Conv Layers

Increasing the number of convolutional layers in the architecture is equivalent to increasing the complexity, or number of trainable parameters in the model. However, there is not a linear relationship between the complexity and accuracy since an over-complex model causes to the problem of overfitting. Therefore, it is crucial to choose it to the correct number. For a summary of the controlled experiments on how

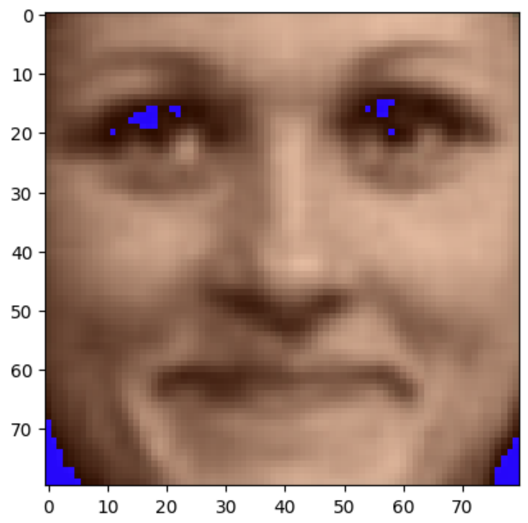

Figure 1: A noisy output prediction example.

number of convolution layers is picked, one can see Table 1 In this context, the performance is observed to increase with increasing number of convolutional layer up to 4 layers, since model is still not complex enough to overfit the data. Note that the learning rate used in the trainings listed in the table is 0.01, since it gives the best result for all 4 trainings.

Table 1: Effect of Number of Conv Layers

| Conv Layer Count | Filter Count | Num. of Epochs | Accuracy |
|---|---|---|---|
| 1 | 8 | 17 | 0.69 |
| 2 | 8 | 27 | 0.7 |
| 3 | 8 | 15 | 0.72 |
| 4 | 8 | 23 | 0.73 |

## 1.2 Effect of Number of Kernels

Increasing the number of kernels is another way of increasing the complexity. It increases the depth of the tensors which is the result of the conv layers. However, the effect of number of kernels depends on the number of convolutional layers too. For example, increasing the number of kernels may not affect the performance with number of conv. layers = 2 whereas it has a crucial impact with number of conv. layers = 4. Therefore, it is important to note the impact with changing the other parameters. The effect is shown in Table 2.

Table 2: Effect of Number of Kernels

| Conv Layer Count | Filter Count | Num. of Epochs | Accuracy |
|---|---|---|---|
| 4 | 2 | 24 | 0.7 |
| 4 | 4 | 20 | 0.72 |
| 4 | 8 | 23 | 0.73 |

### 1.3 Effect of Learning Rate

The learning rate is a critical parameter in machine learning model training. It determines the step size during optimization, impacting convergence speed, stability, and generalization. Selecting an optimal learning rate requires finding a balance between faster convergence and model stability. Careful tuning, often through methods like grid or random search, is necessary for achieving optimal performance. A non-optimal training curve is given below in Figure 2. As can be easily observed, the optimizer cannot optimize the loss curves up to the 5th epoch. After that, there is a sharp update of parameters that leads a fluctuation on the curve. This shows that the learning rate is too large, not optimal. Another situation is shown below in Figure 3. With this example, we conclude that a very low learning rate is not good since the updates lead the optimizer to converge local minima points, which is not the best solution. Therefore, model underfits the data in this case. As a result of the controlled experiments done, the optimum learning rate is found to be 0.01. These results are demonstrated in Table **??**.

Table 3: Effect of Learning Rate

| Conv Layer Count | Filter Count | Learning Rate | Num. of Epochs | Accuracy |
|---|---|---|---|---|
| 4 | 8 | 0.001 | 24 | 0.65 |
| 4 | 8 | 0.01 | 20 | 0.73 |
| 4 | 8 | 0.1 | 23 | 0.72 |

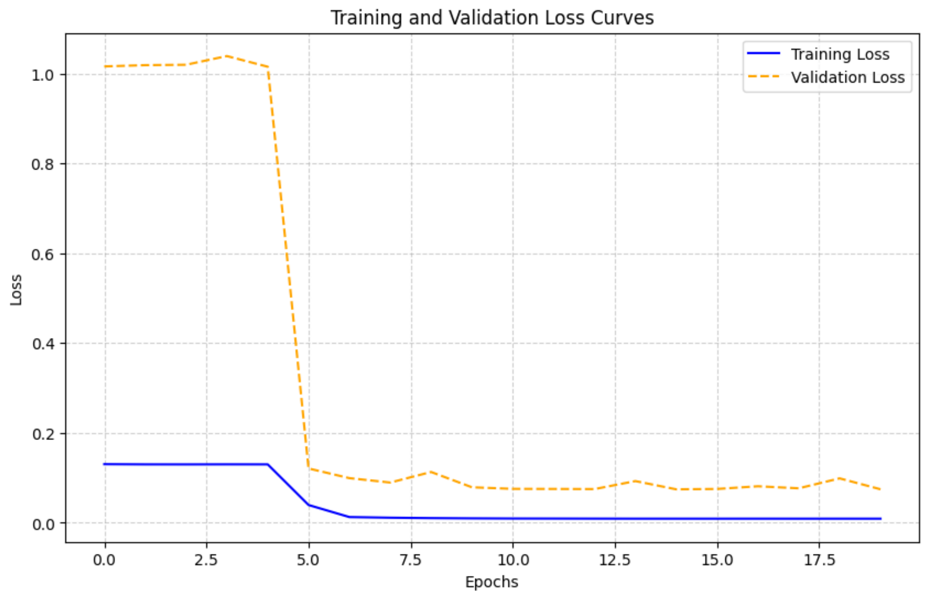

Figure 2: A bad training curve example stemming from a high learning rate choose. (0.1)

## 2 Further Experiments (20 pts)

### 2.1 Adding Batch Normalization

BatchNorm normalizes the inputs to each layer by computing mean and standard deviation within mini-batches. It introduces learnable parameters for scaling and shifting the normalized values, providing the network with flexibility. BatchNorm stabilizes training, allows for higher learning rates, reduces dependency on weight initialization, and acts as a form of regularization.

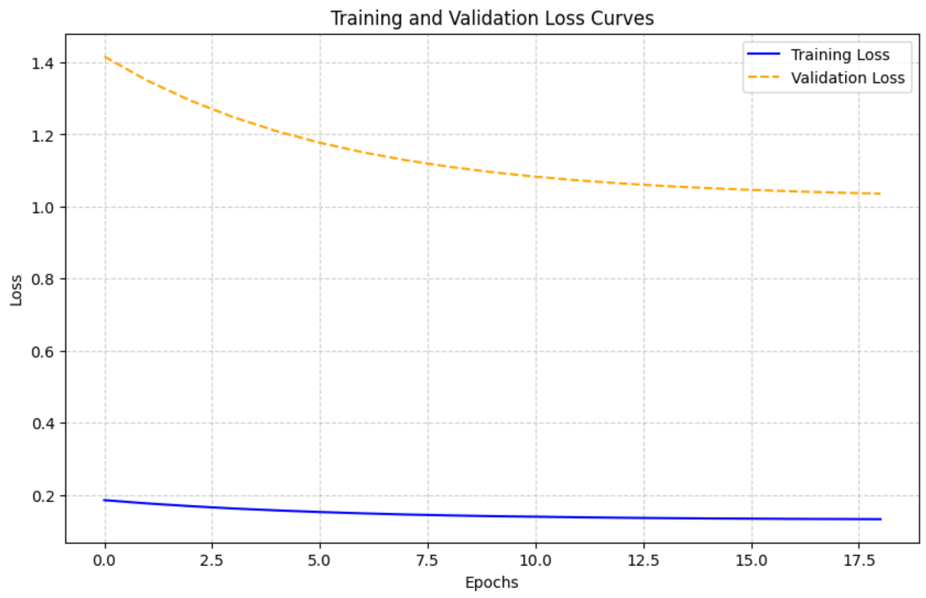

Figure 3: A bad training curve example stemming from a low learning rate choose. (0.0001)

In my test setup, I used the model which has the largest number of trainable parameters, 4 conv layers and 8 num. of kernels. In this setting, adding the BatchNorm layer had a bad impact on the model performance. Since model architecture is not very much complex, adding the BatchNorm caused an underfitting on the training data. Therefore, the accuracy dropped. After trying the other settings such as different number of kernels etc., the accuracy has dropped again, as shown in Table 4

Table 4: Effect of Batch Norm

| Conv Layer Count | Filter Count | Acc. without BatchNorm | Acc. with BatchNorm |
|---|---|---|---|
| 4 | 8 | 0.73 | 0.54 |
| 4 | 4 | 0.72 | 0.55 |
| 3 | 8 | 0.72 | 0.50 |

## 2.2 Adding Tanh Activation Function

In summary, the tanh activation function squashes input values to the range of (-1,1) providing a zero-centered output with desirable properties for optimization.

In the IC case with the above mentioned candidate architecture parameters, adding the tanh function at the very end of the model did not have a major impact on the accuracy of the model. However, the noise demonstrated in the Figure 1 is decreased when it is tested visually. Therefore, it is preferred to be used. This is an expected result because tanh maps the inputs to the output range (-1,1) as explained before. Therefore it makes sense to use this activation function at the very end of the architecture.

## 2.3 Increasing the Number of Channels

Increasing the number of channels to 16 slightly increased the accuracy on the validation dataset. Therefore, I keep it in my architecture. (Accuracy increased to 0.74 from 0.73)

## 3 Your Best Configuration (20 pts)

The choose of the best configuration is done using the performance metrics explained in section 1. Among those performance metrics, the most important one is the accuracy, since it is the final indicator of how well the model is coloring the individual pixels. However, this is a final result of the other metrics too, such as the smoothness of the learning curve and quality of the training process.

In addition to the choose of the best architecture and training parameters, the maximum number of epochs the model should be trained is another crucial decision to make in this process. For this decision, an early stopping algorithm is used. This early stopping algorithm has two input arguments, namely patience and improvement threshold. With this two inputs, algorithm works as follows: Stop the training if an improvement higher than the improvement threshold has not been observed throughout the last 'patience' layers. By using this algorithm, the plateau which the learning has completed is detected and training is stopped.

With the above mentioned algorithm, the automatically decided number of epochs is 22. In addition to the number of epochs, there are many more decisions taken. For the number of convolutions, as mentioned above, 4 is picked since it outperforms the other candidates in the controlled experiments made. With this choose, number of kernels are experimented too, as mentioned again in Section 1.2 and it decided to be 16. In addition, the tanh is used at the very end of the architecture to prevent output noises.

After deciding the architectural parameters, the training settings are made. In this part, various numbers for learning rate is examined and the best one was 0.01 for the specific architecture. With these decisions, the visual and numeric information is provided below.

### 3.1 Learning Curves

The plot of the training and validation mean-squared error loss over epochs is provided below in the Figure 4.

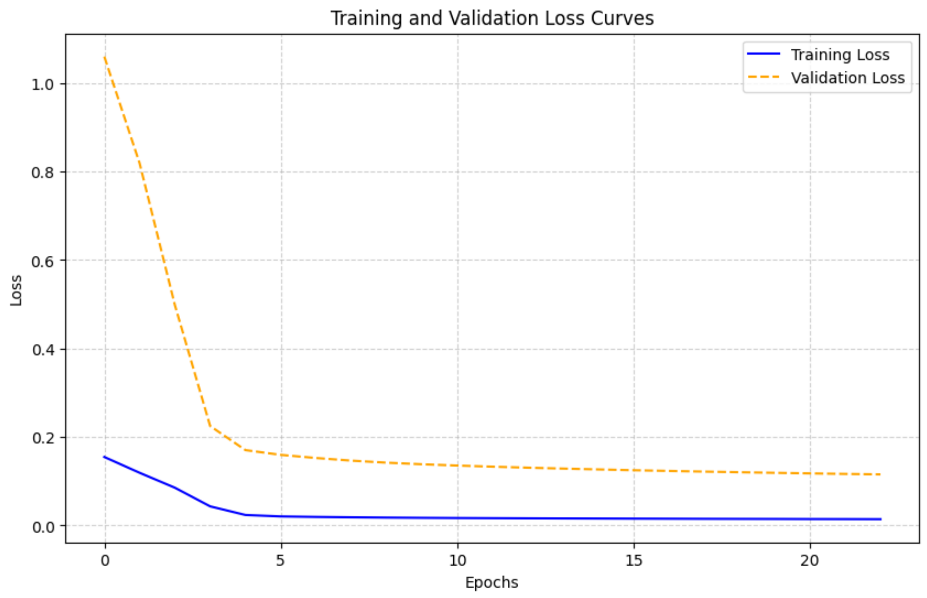

Figure 4: Training curves of the best configuration.

### 3.2 Validation 12-margin Error

The plot of the validation 12-margin error versus epochs plot is provided below in Figure **??**.

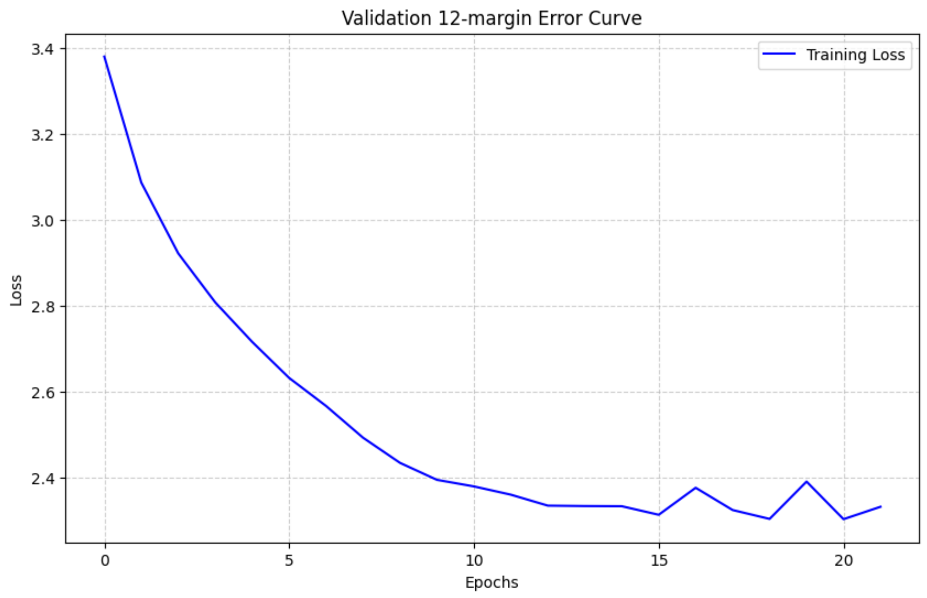

Figure 5: Validation 12-margin error curve.)

## 3.3 Prediction versus Ground Truth Examples

There are 5 examples of predictions versus ground truth color images below.

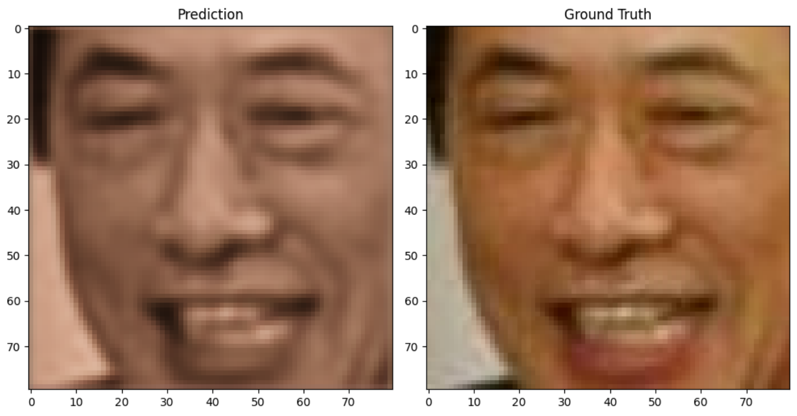

Figure 6: Demonstration 1

## 3.4 Advantages and Disadvantages of the Model

This model has the advantage of simplicity firstly. Due to it's non-complex architecture and the its low number of parameters, the model make its predictions fast, completing its work in a limited time. In addition, with the use of the tanh activation function at the very end, which maps its inputs to the range -1,1, the noises at the output is avoided and a smooth and visually good output is obtained.

When it comes to the disadvantages of the model, the most crucial problem is the accuracy. The model does not work accurately since it is not complicated enough in some sense. This is the one single main drawback of the model. The non-complex nature of the model can also be understood through the observation of the learning curves which are not ideal.

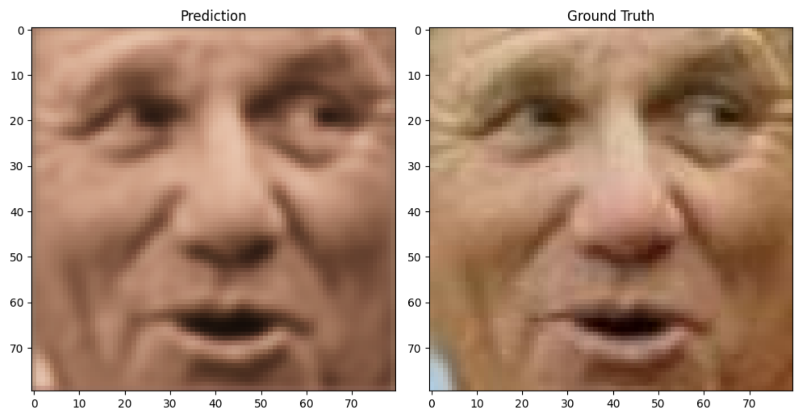

Figure 7: Demonstration 2

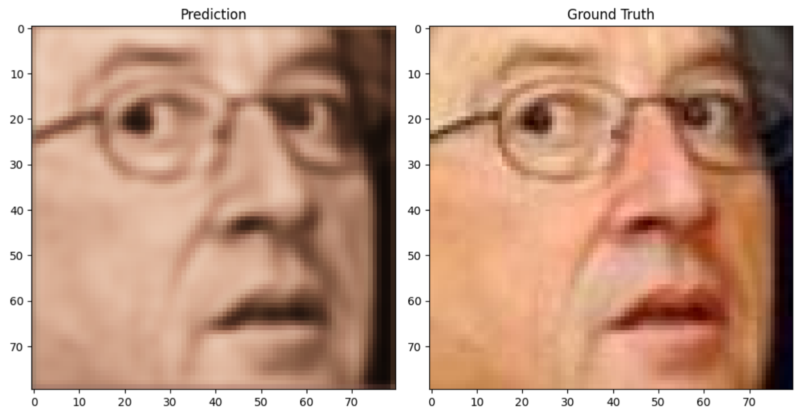

Figure 8: Demonstration 3

## 4 Your Results on the Test Set (30 pts)

In case of a problem, you should run template.py file, which I edited a little bit for demonstration purposes. This python script first trains the model with my best configuration settings, and then save the estimations and image names text file. For the script to test the model on the test set, you should change the folder path variable to the path of the test set. Thus, you can directly access my estimations on the test set.

## 5 Additional Comments and References

This Take Home Exam was not only a homework but also a tutorial for us too. Trying to improve the performance of a convolutional neural network, learning how to analyse the output and the training process was crucial in terms of gaining experience in model training and using PyTorch library, which is the mostly used library in both computer vision and machine learning applications. For this work, I want to thank those who contributed.

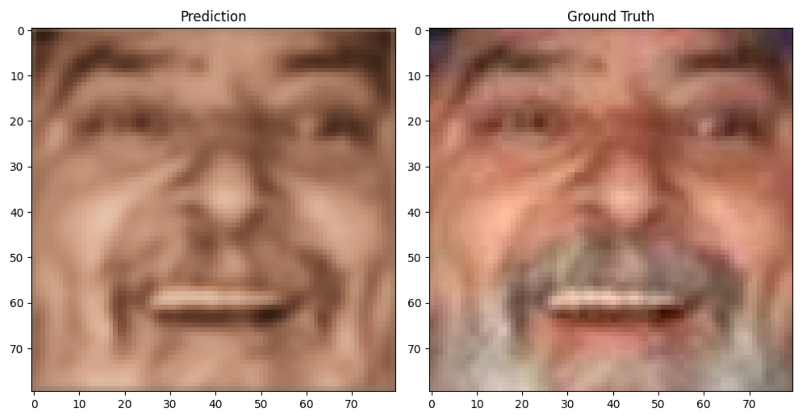

Figure 9: Demonstration 4

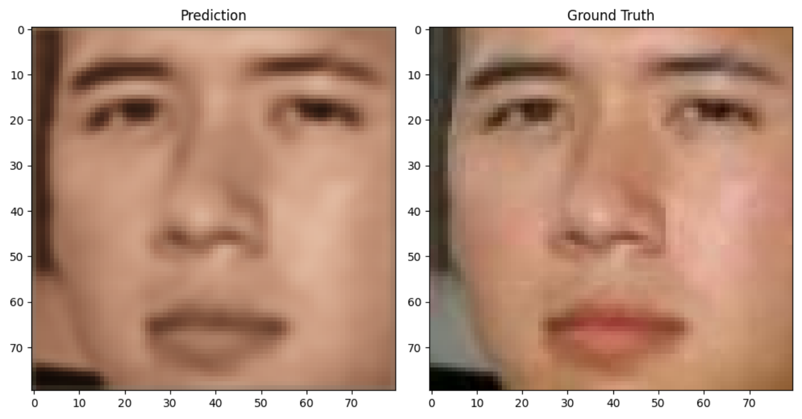

Figure 10: Demonstration 5