CENG 483

Introduction to Computer Vision

Fall 2022-2023

Take Home Exam 3

Image Colorization

Student ID 1: 2448587

Student ID 2: 2380004

1 Baseline Architecture (30 pts)

We used our course instructor's videos for the information[2].
Some hyperparameters has the same value throughout the experiment. These are:
- Batch size: 16
- Stride: 1
- Padding: 1
- Bias: True

- Discuss effect of the number of conv layers:
  Other hyperparameters are; number of kernels: 4, learning rate: 0.1

(a) Layer:1, Acc: 0.719/1.00

(b) Layer:2, Acc: 0.730/1.00

(c) Layer:4, Acc: 0.712/1.00

The code automatically chooses epoch number. And in this example it was 30. In (a), it seems that the model with only one convolutional layer has a lower accuracy compared to (b). In (b) Layers:2, it seems that the model with two layers has a better accuracy overall. In (c) Layers:4, it seems that the model with 4 layers has a lower accuracy compared to others.

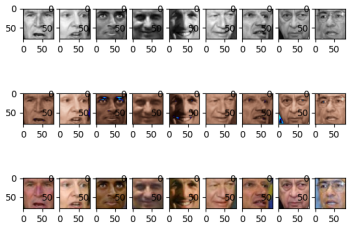
It's worth noting that the accuracy alone doesn't give a full picture of the performance of the model. Additionally, the accuracy of the model may not be a good indicator of its performance on unseen data. Other than that, images in (b) are better than (a) images. However, these 9 images do not really represent the overall situation.

Adding more convolutional layers makes the model more complex. Hence, it can have more parameters. It is good to have parameters, but while adding more layers one should consider overfitting. In our situation 2 layers were enough to get good result.

- Discuss effect of the number of kernels(except the last conv layer): Other hyperparameters are; convolutional layers: 4, learning rate: 0.1

(a) Kernel:2, Acc: 0.622/1.00

(b) Kernel:4, Acc: 0.712/1.00



(c) Kernel:8, Acc: 0.646/1.00

The code automatically chooses epoch number. And in this example it was 24. Best configuration was (b) where number of kernels is 4. It is followed by (c) where number of kernels is 8. Finally, the worst result was (a) where number of kernels is 2. This hyperparameter states the out channels of the first convolutional layer. In our training, (a) and (c), num. of kernels 2 and 8 gave similar results. Training (b) was the best with %71.2 accuracy. The number of kernels, also known as the number of filters or output channels, controls the number of learned features that the model can extract from the input data. The number of kernels determines the depth of the output feature map, which is the number of channels in the output of the convolutional layer. Increasing the number of kernels allows the model to learn more complex representations of the input data, which can lead to better predictions. Each kernel performs a convolution operation on the input data, and the output of each kernel is a feature map. More kernels means more feature maps and more learned features. In our case, middle value gave the best result.

- Discuss effect of the learning rate by choosing three values: a very large one, a very small one and a value of your choice:
  Other hyperparameters are; number of kernels: 4, convolutional layers: 2

(a) LR:0.1, Acc: 0.728/1.00

(b) LR:0.01, Acc: 0.713/1.00

(c) LR:0.0001, Acc: 0.554/1.00

Best configuration was (a) where learning rate 0.1. It is followed by (b) where learning rate is 0.01. Finally, the worst result was (a) where learning rate is 0.0001. Accuracy in training (c) was really smaller than other cases. Because, this value is not enough to converge in 100 epochs.
Additionally, we tried to train the model with a learning rate=1000. However, we get nan as loss values. As a result, we did not get a proper model.

2 Further Experiments (20 pts)

- 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.

(a) Without batchnorm

(b) With batchnorm

Figure 1: Acc: 0.730/1.00 → 0.733/1.00

Batch normalization is beneficial in CNNs because it helps to stabilize the training process and improve the accuracy of the model. It does this by normalizing the activations of each layer, which can reduce the internal covariate shift and make the training process converge faster. Additionally, batch normalization can also act as a form of regularization, which can further improve the model's performance.
As a result, it increased the accuracy of validation set for our model. So we keeped it.

- 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.

(a) Without tanh

(b) With tanh

Figure 2: Acc: 0.733/1.00 → 0.720/1.00

The tanh function is a non-linear activation function that maps the input to the range [-1, 1]. It is differentiable and computationally efficient.

It can help to improve the model's performance by introducing non-linearity like ReLU and allowing the model to learn more complex representations of the input data. In our case it didn't.

One of the advantages of using the tanh function is that it can help to prevent the vanishing gradient problem[3] , which is a common issue in deep neural networks where the gradients become very small during backpropagation, making it difficult for the model to learn. Another advantage is that it can be a better alternative to sigmoid function[1], since it can saturate at extreme values.

In our case, tanh function did not give better results and it decreased accuracy. This could be because of overcomplicating the model. We did not keep it.

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

(a) With channel size 4

(b) With channel size 16

Figure 3: Acc: 0.733/1.00 → 0.735/1.00

Increasing the number of channels in a CNN can increase the model's performance because it allows the model to learn more complex representations of the input data. Each channel in a convolutional layer acts as a separate feature detector, and having more channels means that the model has more feature detectors to work with. This can enable the model to pick up on more subtle features in the input data, which can lead to better performance on the task at hand. Additionally, having more channels can also increase the capacity of the model, which can help it to better fit the training data and improve its generalization capabilities.
As a result, it increased the accuracy of validation set for our model. So we keeped it.

- Try replacing conv layers with **separable conv** or **group conv** layers. How does it affect the results, and, why? Keep it if it is beneficial. **Note that** number of kernels in separable conv and group layers should be the same as previously selected architecture.

  We tried to change group parameter on Conv2d, however it must be a divider of both in_channel and out_channel number. So, we must set it to 1 which is the default parameter that is already used in all layers.

3 Your Best Configuration (20 pts)

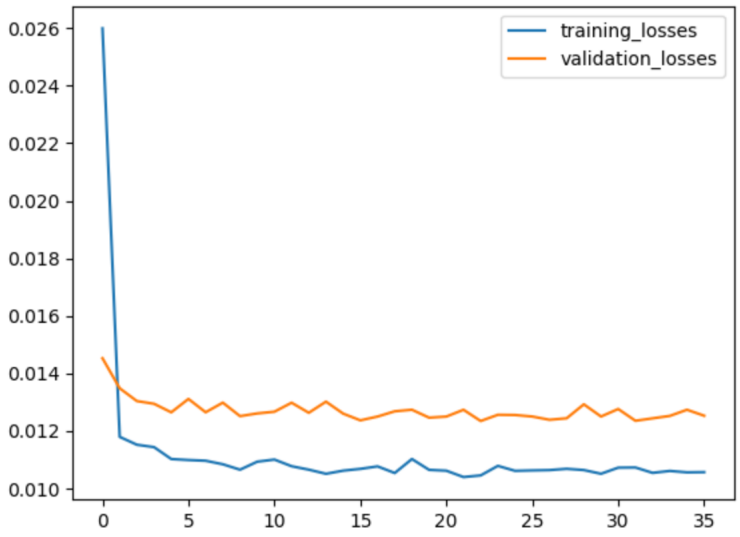
Using the best model that you obtain, report the following:

- The automatically chosen number of epochs(what was your strategy?):
  The automatically chosen number of epochs is 36.

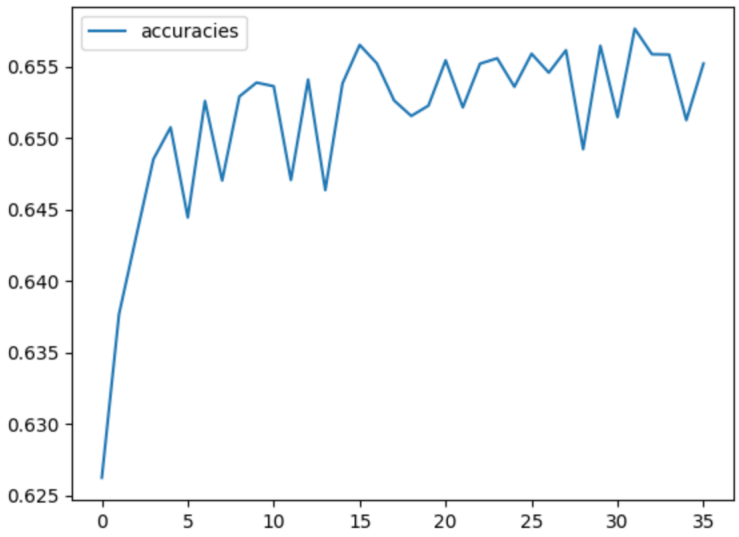
  ```
  if validation_loss < 0.5 or last_validation_loss <= validation_loss + ↩
      0.0001:
       break
  ```

  With this code segment, we tried to achieve that, when the validation loss starts to increase or decrease a little bit, we stop and choose the epoch size. Also, if validation loss is less than a small value like 0.5, epoch number will be automatically chosen.

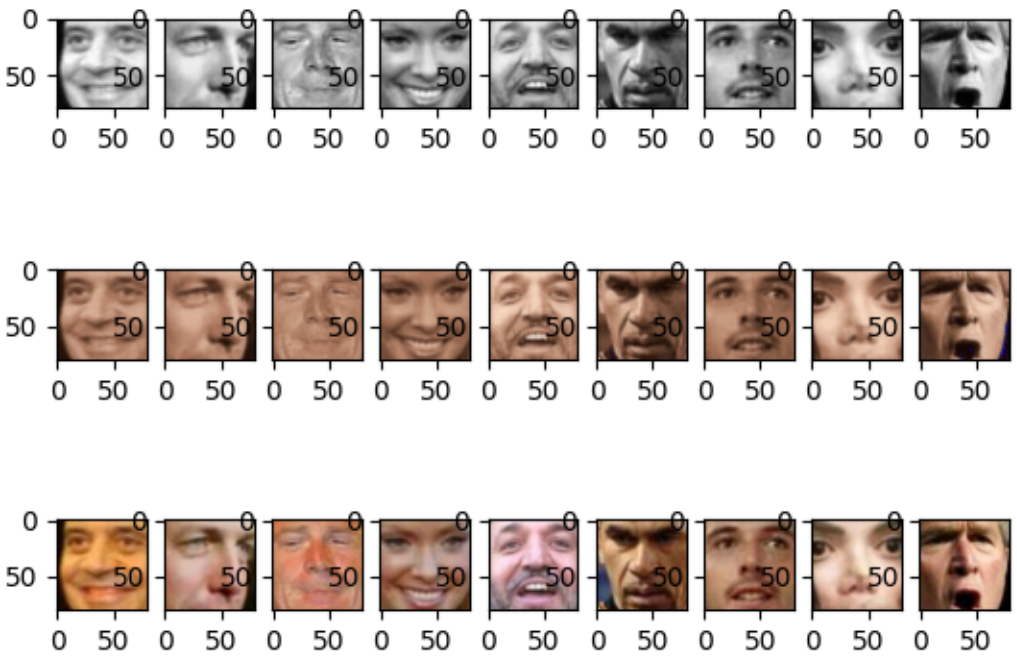
- The plot of the training mean-squared error loss over epochs:



- The plot of the validation 12-margin error over epochs:



- At least 5 qualitative results on the validation set, showing the prediction and the target colored image:



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

  It is a powerful type of neural network that are commonly used in image and video processing tasks. We used for 80x80 images in our case. The model has several advantages:

  - The model is able to automatically learn and extract useful features from the input data, which can lead to better performance compared to traditional hand-crafted features.
  - The model is able to learn spatial hierarchies of features, which allows them to learn both local and global features of the input data. Even if we trained the model with 80x80 images, the model would be able to handle input data of varying sizes and resolutions.

  However, the model also have some disadvantages:

  - The model requires a large amount of labeled data to train effectively, we used 5001 training images with ground truths. It can be costly and time-consuming to obtain.
  - The model can be prone to overfitting because too similar examples were supplied to the model.
  - The model is somewhat computationally expensive, with a 2,3 GHz 8-Core Intel Core i9, it took 5-10 mins for high epoch cases.

  Potential ways to improve the model would be:

  - Using data augmentation techniques to increase the amount of training data and improve the model's ability to generalize.
  - Using techniques such as attention mechanisms or self-supervised learning to learn more robust features from the input data.
  - Using techniques such as pruning or quantization to reduce the number of parameters and computational cost of the model.

4 Your Results on the Test Set(30 pts)

4.1 Usage with create_npy.py

```
$ rm test_images.txt
$ python create_npy.py
$ python evaluate.py estimations_test.npy test_images.txt
```

4.2 Usage with supplied estimations_test.npy

```
$ python evaluate.py estimations_test.npy test_images.txt
```

4.3 In case of error in create_npy.py

After running create_npy.py; if there is an error after the training of the model, follow these steps:

1. Comment the main loop part in the cnn.py.
2. Correct the parameters in the create_npy.py
3. Run create_npy.py again.

This will solve the problem, after that evaulation can be made.

```
197  device = torch.device(DEVICE_ID)
198  print('device: ' + str(device))
199  # ---- main loop -----
200  for num_of_kernel in num_of_kernels:
201      for learning_rate in learning_rates:
202          for conv_layer_number in conv_layer_numbers:
203              net = Net(conv_layer_number=conv_layer_number, num_of_kernel=num_of_kernel, tanh=False, batch=True).to(
204                  device=device)
205              net.run(learning_rate=learning_rate)
```

Figure 4: Main loop part

5 Additional Comments and References

References

[1] Activation Functions: Sigmoid vs Tanh. Baeldung. https://www.baeldung.com/cs/sigmoid-vs-tanh-functions

[2] Cimbiş, R. G. (2021-2022). Ramazan Gökberk Cimbiş [Video file]. Retrieved from https://www.youtube.com/playlist?list=PL-8aJZY3HfAQbiSb3FdisSnProFfxhqjc

[3] Wang, C. (2021, December 7). The Vanishing Gradient Problem - Towards Data Science. Medium. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[4] Welcome to PyTorch Tutorials — PyTorch Tutorials 1.13.1+cu117 documentation. (n.d.). https://pytorch.org/tutorials/