CENG 483

Introduction to Computer Vision
Fall 2023-2024

Take Home Exam 3: Image Colorization

Full Name: Meriç Karadayı
Student ID: 2448553

1 Baseline Architecture (30 pts)

To find the best baseline architecture, I have used grid search method for hyperparameters to train the model. After all trainings are completed, I have chosen the fixed values whose graphics and results are showing the effect of the changing parameter more obvious.

1.1 Effect of the number of Conv Layers

I have done many experiments to observe the effect of convolution layer numbers. Some experiments I have carried are discussed below.

1.1.1 Number of Kernels: 2 - Learning rate: 0.01

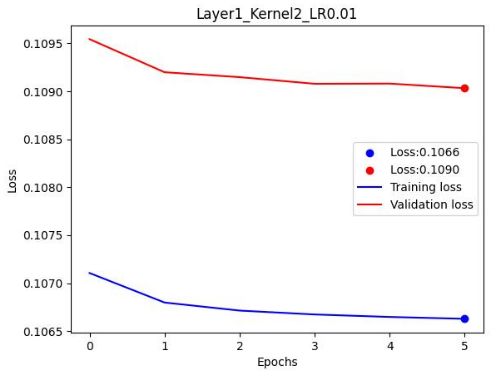


Figure 1: 1-Conv Layer results

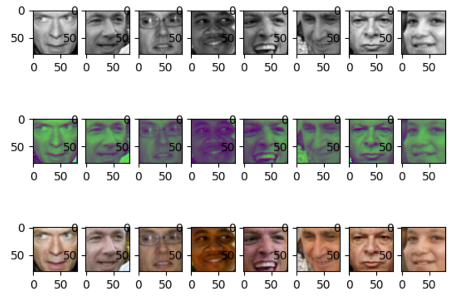


Figure 2: 1-Conv Layer images

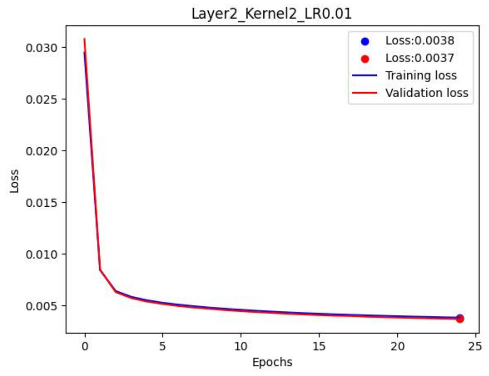


Figure 3: 2-Conv Layer results

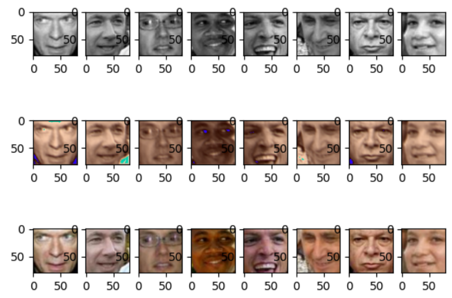


Figure 4: 2-Conv Layer images

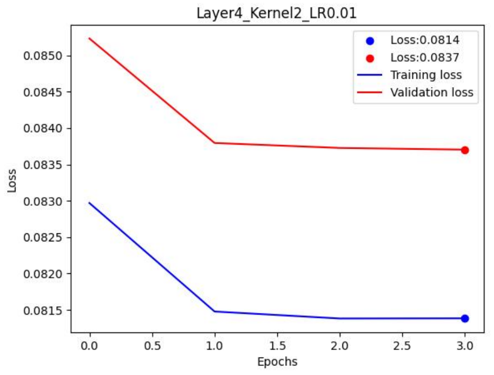


Figure 5: 4-Conv Layer results

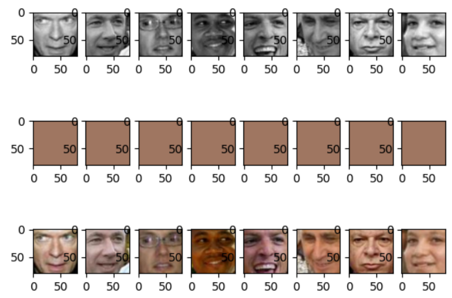


Figure 6: 4-Conv Layer images

In this experiment I have fixed the number of kernels to 2 and the learning rate to 0.01. The model with 2 convolution layers has outperformed the models with 1 and 4 layers. I have added an early stop to my training. 1 and 4-layered models were stopped way earlier than the 2-layered. When the plots are inspected, we can see that the final loss on the validation set is also have a significant difference. Furthermore, 4-layered has stopped in the 3rd epoch which is my patience for early stop, 4-layered was barely learning with this configuration. For these parameters, we can say that both cases where the number of layers is small and big do not contribute to the model. A proper optimum parameter should be found and set.

In addition, we can also observe that the loss value does not represent totally how well our model predicts. The loss of 4-layered model is less than the 1-layered but the reason of such difference is that, layered is full of skin color and this leads to smaller loss that the 1-layered.

1.1.2 Number of Kernels: 4 - Learning rate: 0.01

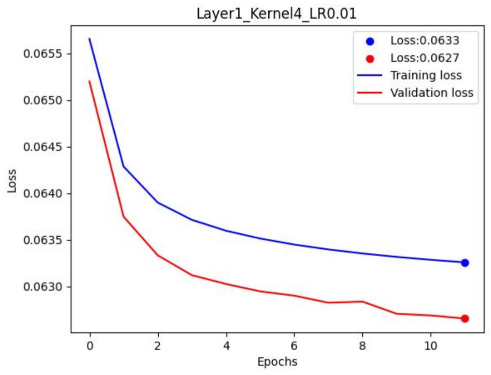


Figure 7: 1-Conv Layer results

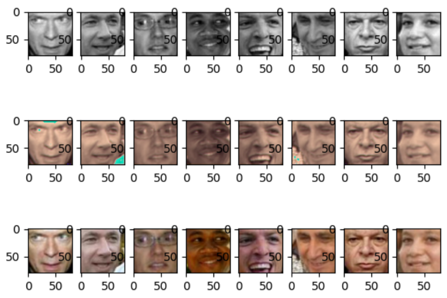


Figure 8: 1-Conv Layer images

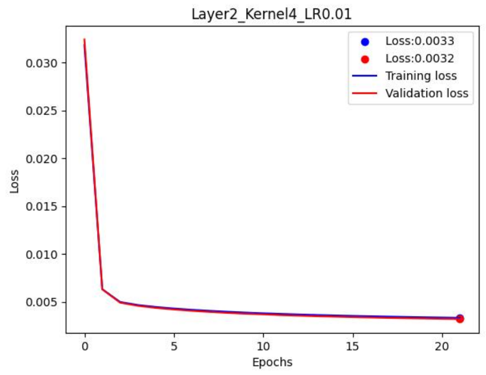


Figure 9: 2-Conv Layer results

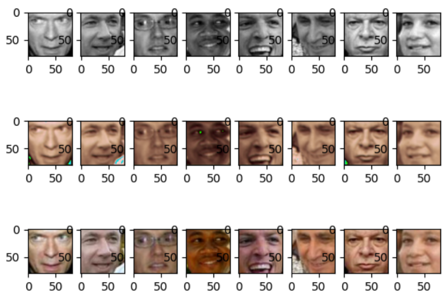


Figure 10: 2-Conv Layer images

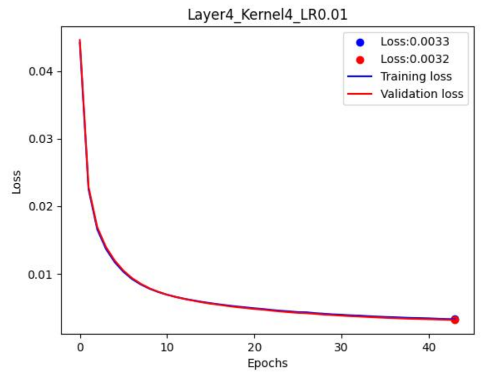


Figure 11: 4-Conv Layer results

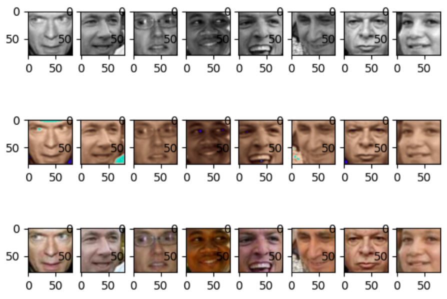


Figure 12: 4-Conv Layer images

In this experiment, I have fixed the number of kernels to 4 and the learning rate to 0.01 and wanted to see whether the effect of the number of layers will remain the same in the previous experiment. However, when the losses of trained models on validation set are compared, 4-layered and 2-layered are both performing well. Moreover, while the slope of the 2-layered is decreasing sharply, 4-layered is decreasing more smoother which is more desired plot slope. Unlike the previous example, 4-layered is not being outperformed by 2-layered. However, if we have also trained a model with more than 4 layers, I think the performance of the model could get worse according to my observations on the first experiment.

1.1.3 Conclusion

I think the number of layers is highly dependent on the other parameters. It does not have a straightforward effect on the results such as increasing or decreasing layer size is increasing the performance of the model. It has an effect on learning but this parameter should be finetuned specifically for that case and specific the other parameters. To conclude, generalization of the effect of number of layers is not a good approach, increasing or decreasing this value can harm the model and the optimum value should be found.

1.2 Effect of the number of number of Kernels

1.2.1 Num. of Layers: 2 - Learning Rate: 0.001

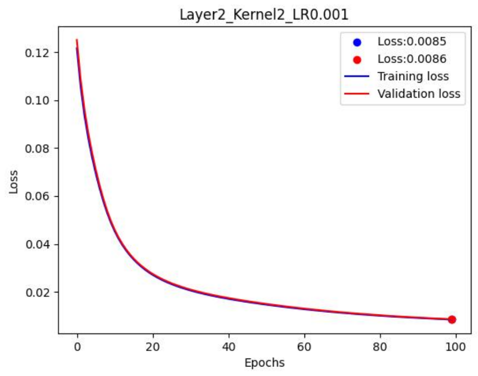


Figure 13: 2-Kernels results

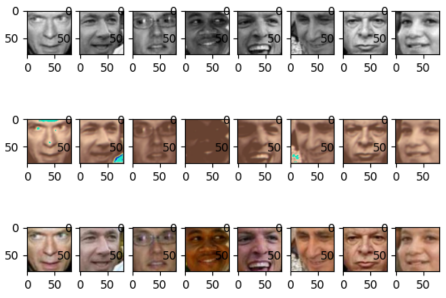


Figure 14: 2-Kernels images

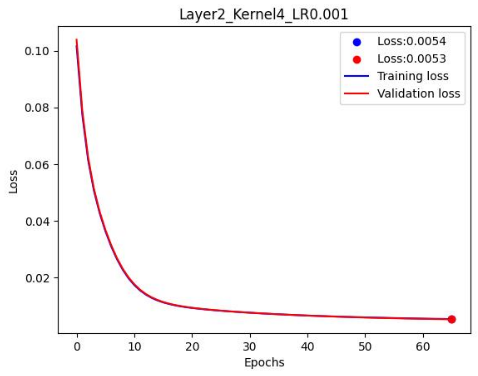


Figure 15: 4-Kernels results

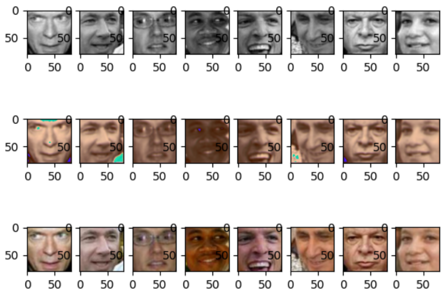
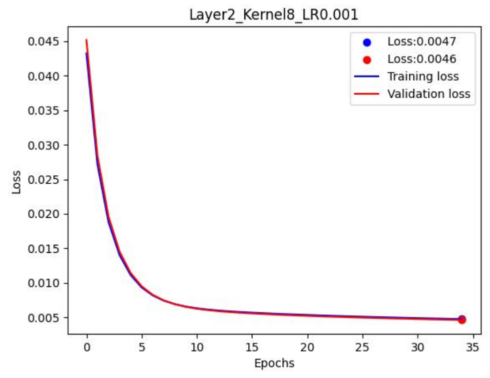


Figure 16: 4-Kernels images



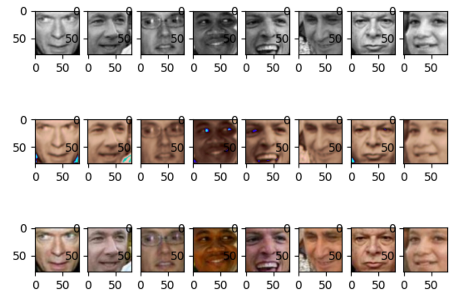


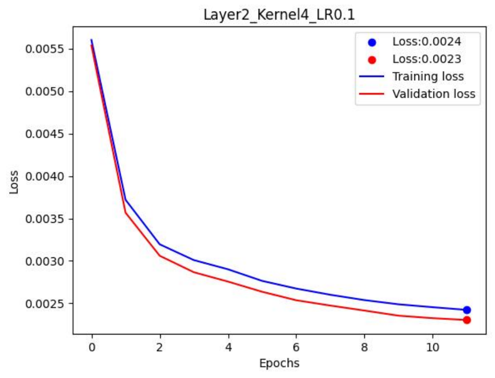
Figure 17: 8-Kernels results

Figure 18: 8-Kernels images

In this experiment, I have used 2 layers and 0.001 learning rate for training and tried 2, 4, and 8 Kernels in my models. When their graphs are examined, we can say that all 3 give good loss slops. They are not changing sharply or they do not stay constant. However, their final loss results in the validation set vary from each other. The loss of 2-Kernel is 0.0086, 4-Kernel is 0.0053, 8-Kernel is 0.0046. We can see that increasing the number of kernels decreases the total loss. The kernel size in layers is basically the channel sizes our kernel has, so more kernel means more weight and more possible parameters to learn. Therefore, increasing the number of weights could be the reason for this positive effect on our model.

1.3 Effect of Learning Rate

1.3.1 Num. of Layers: 2 - Num. of Kernels: 4



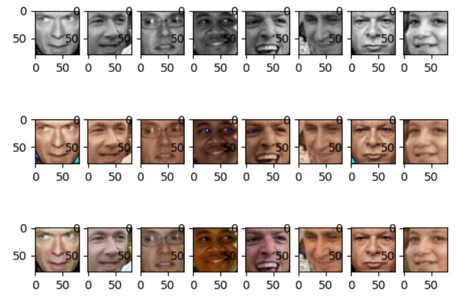
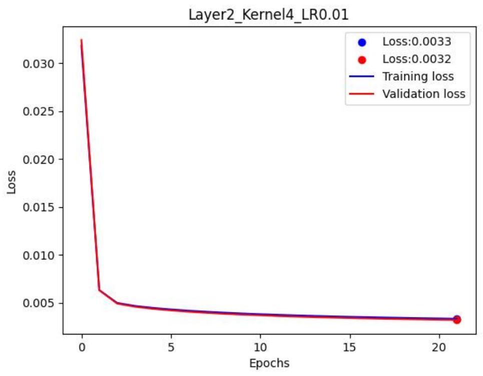


Figure 19: 0.1 Learning Rate results

Figure 20: 0.1 Learning Rate images



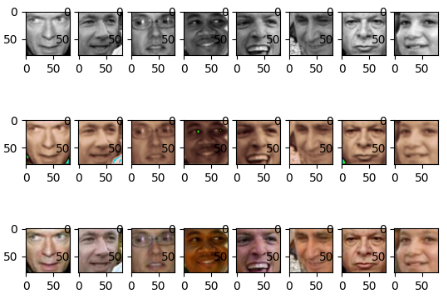
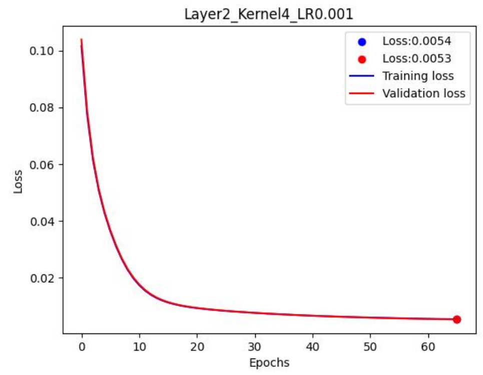


Figure 21: 0.01 Learning Rate results Figure 22: 0.01 Learning Rate images



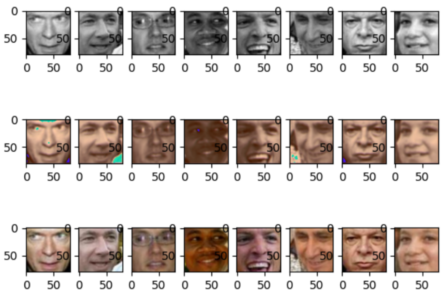
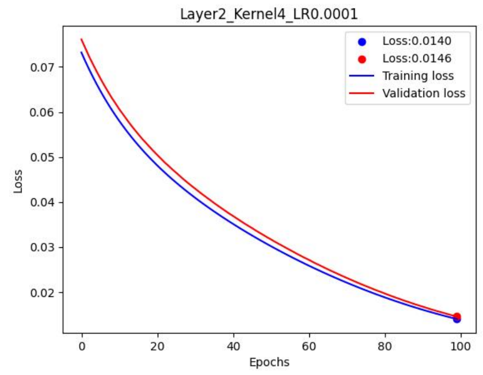


Figure 23: 0.001 Learning Rate results Figure 24: 0.001 Learning Rate images



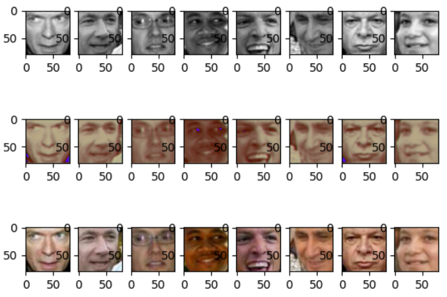


Figure 25: 0.0001 Learning Rate results Figure 26: 0.0001 Learning Rate images

In this experiment, I have used 2 convolution layers with 4 kernels in each layer. Change of learning rate changed the number of epochs in each run. Increasing learning rate decreases the number of epochs and decreasing it increases the number of epochs as expected. Using a small learning rate such as 0.0001 causes the model to not converge. It has run 100 epochs but its graphs show that it did not converge, also the fact that its loss is 4-5x bigger than others shows that choosing a small learning rate could be a bad choice. 0.1 learning rate has the smallest loss but it has converged so quickly. Even though it is not visible from the figure since the first value is shown after the first training, we can see from other figures that the initial loss is about 0.1. The decrease in loss is so sharp while using this learning rate, therefore I think this could be a problem.

2 Further Experiments (20 pts)

I have observed my best plot with the following parameters even though it does not have minimum loss among the results. It was the most smooth graphic with a low loss value. Some of the models with 0.1 learning rate were performing lower losses than this configuration but were decreasing so sharp.

- Num. of Conv Layer: 2
- Num. of Kernels: 8
- Learning rate: 0.001

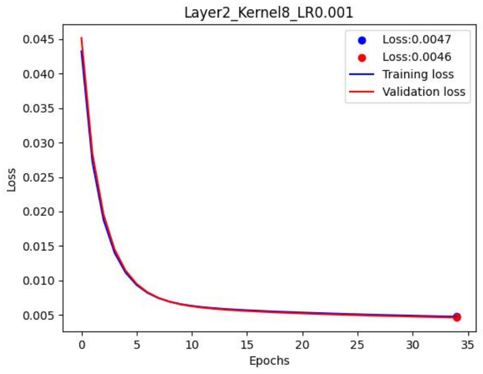


Figure 27: Chosen model results

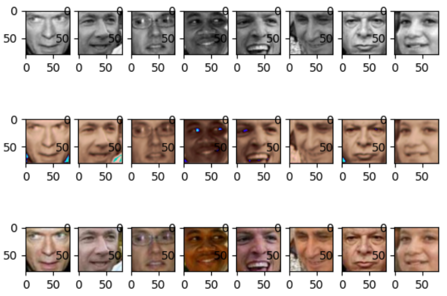


Figure 28: Chosen model images

2.1 Effect of Adding Batch Normalization

Using batch normalization accelerated the training but it also increased my loss on validation set from 0.0046 to 0.0059. I think that the decrase of time does not compensate such loss increase. Therefore I am not going to use it.

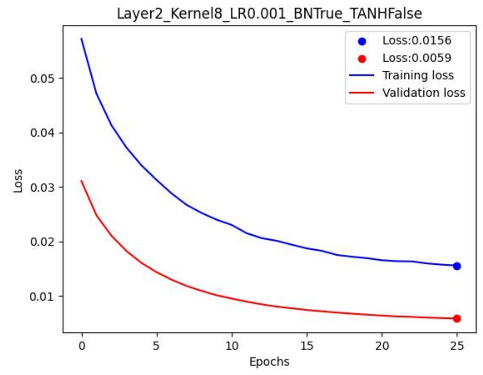


Figure 29: Using batch normalization results

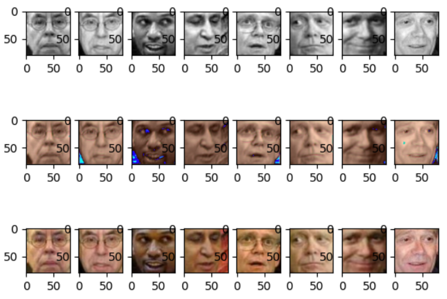


Figure 30: Using batch normalization images

2.2 Effect of adding Tanh

Using tanh after the last layer has changed the model in many aspects such as the number of epochs, loss value, and image results. Let's start with the epoch number. It has decreased significantly. It was being trained in approx. 35 epochs in the previous experiment, however, training this model lasted approximately 15 epochs. It has learned faster and converged sooner than the previous experiment but its learning curve seems much proper. I believe that this is a positive effect. The second change we are observing is that the tanh decreased the loss from 0.0046 to 0.0042. This is also a positive effect. The third change is the visual change in outputs. I think this is the most obvious and most important feature it tanh is adding to our model. When images in figure 30 and images in figure 32 are compared. We can clearly see that the tanh has removed some blue and green points on outputs that should not already exist. After all 3 positive effects I will add it to my structure. The predictions model made are now way more natural than previous experiments.

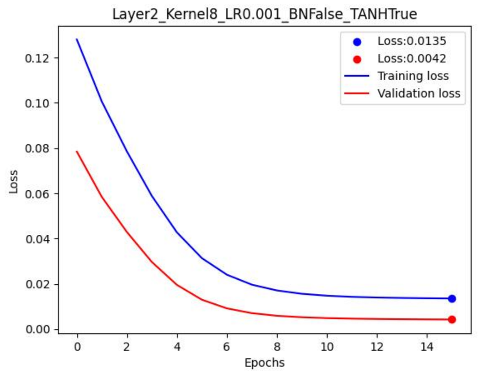


Figure 31: Using tanh results

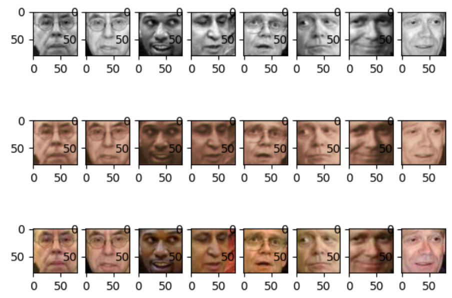


Figure 32: Using tanh images

2.3 Effect of increasing the number of channels to 16

We have already discussed and experimented with the effect of change in the number of kernels in section 1.2. We have concluded that increasing the kernel number improves our model and decreases the loss. In this part, I have used 16 kernels instead of 8 kernels. The results in this part are not consistent as we derived from our previous experiments. The loss has increased from 0.0042 to 0.0047. This reason of this inconsistency can be due to our inputs and network size. Probably for larger inputs and networks using 16 kernels would give better outputs than 8 kernels, but our case 16 kernels does not benefit. Therefore I am not going to use it.

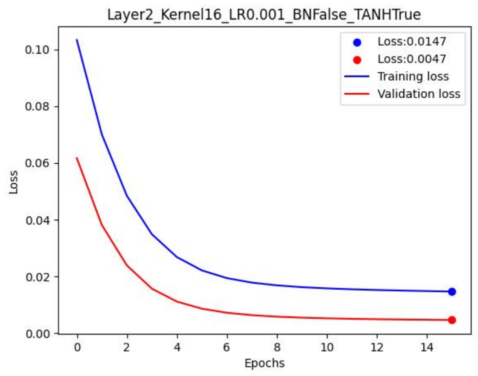


Figure 33: Channel size 16 results

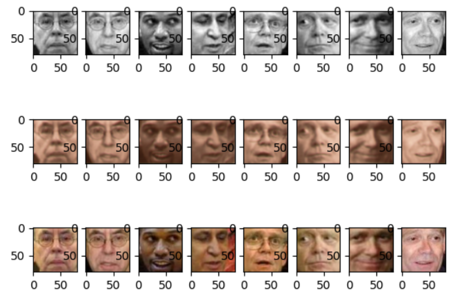


Figure 34: Channel size 16 images

3 Your Best Configuration (20 pts)

The best model I have obtained have the following configurations

- Number of layers: 2
- Number of kernels: 8
- Learning rate: 0.001
- tanh after last layer, ReLU after other layer

3.1 Number of Epochs

The automatically chosen number of epochs (what was your strategy?): For choosing epochs, I have implemented an early stop. This concept is doing a simple comparison between the current loss and previous losses. It works as follows. the model is saving the minimum amount of loss between the epochs. After each epoch, the loss of that epoch and the minimum loss are compared with each other. If the current loss is only bigger than minimum by a predefined small value (early stop delta hyperparameter) 3 times in a row, the model stops training. This ensures model to stop learning if the change in loss is too small and do not improve the result much. By doing this the training time is being reduced significantly.

3.2 Final results on validation set

I have trained my model using mean-squared error as loss function. The loss values of training are calculated with mean-squared loss while the loss values of validation are calculated using 12-margin error on the graphic. The results can differ from the previous results because on previous experiments I have also used 12-margin loss during training.

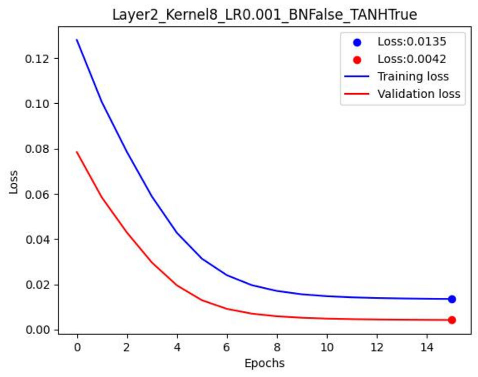


Figure 35: Final results

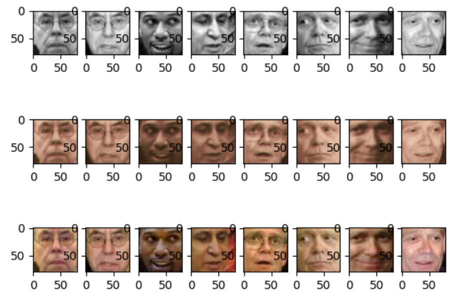


Figure 36: Final images

3.3 Discussion

The model has both strong and weak aspects. The model is trained on a dataset that includes small face images of people. Therefore most of the pixels are in skin color. This is both the strong and weak side of our model. Some images have a background color different than skin color. Since model is trained mostly on faces it is can not predict that parts of the image correctly. The predictions of the model are being again colors similar to skin color this is one problem model is facing. However, we are mostly dealing with the faces, therefore the result in faces is less noisy. In addition, our model is also predicts non-skin colored parts of the face such as eyes and teeth also similar to the face color which prevents our model from predicting more accurately. To improve our model's performance, I think we can made the following operations.

- Now we are randomly initializing our weights, however using more advanced techniques such as Xavier initialization can avoid due to these random weight can generate.
- In current model we are using SGD optimizer in default version which does not present much features. Instead of using such optimizer, we can use more advanced optimizers or we can add features to our optimizer such us weight decay or momentum.
- We have used fixed kernel size 3 for all of our experiments. However tuning this parameter can also decrease our loss.

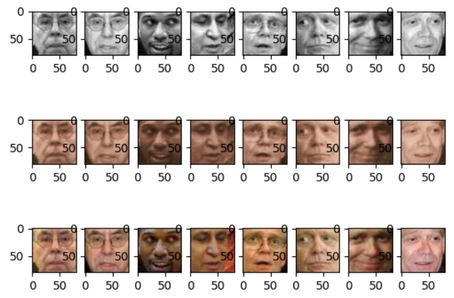


Figure 37: Example results

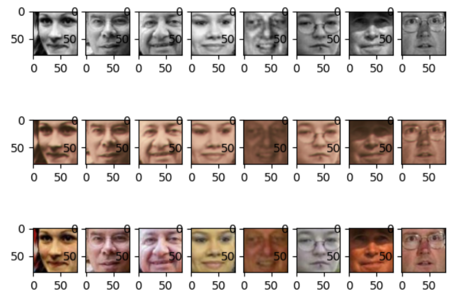


Figure 38: Example results

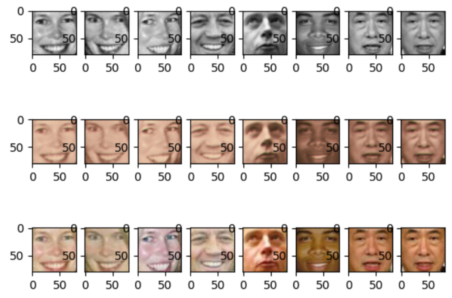


Figure 39: Example results

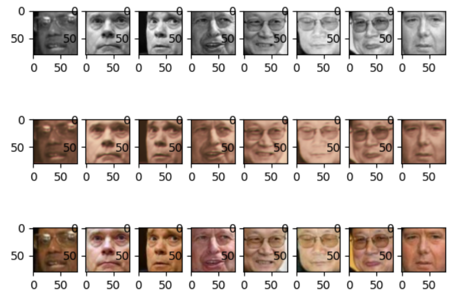


Figure 40: Example results

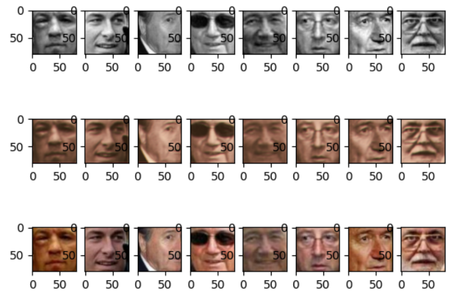


Figure 41: Example results

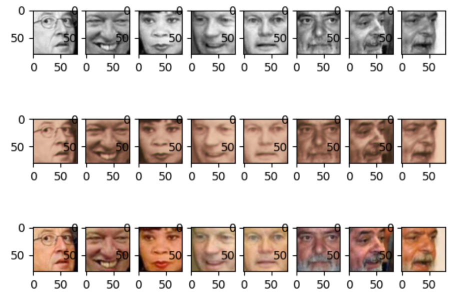


Figure 42: Example results