CENG 483

Introduction to Computer Vision

Fall 2022-2023

Take Home Exam 3

Image Colorization

Student ID: 2010023

Please fill in the sections below only with the requested information. If you have additional things to mention, you can use the last section. Please note that all of the results in this report should be given for the **validation set** by default, unless otherwise specified. Also, when you are expected to comment on the effect of a parameter, please make sure to **fix** other parameters. You may support your comments with visuals (i.e. loss plot).

Preamble

We would like to touch on the approaches followed in experiments, and how we formed our reporting scheme in this paper.

To automatically choose the number of epochs (and iteration also) we utilized casual early stopping with few hyper-parameters like *patience* and *delta (or threshold)*. We used a currently implemented early stopping class from [4]. Here we fixed *patience=3* and *delta=5e-5* for all experiments by default and used it for all experiments reported here unless otherwise specified. We monitor stopping condition and saving a checkpoint if conditions satisfied every 100 batch iterations (following `template.py`) and thus three times per epoch, checkpointing is done only if there's an improvement and not every 100 iterations. A quick note here is that we choose to monitor validation loss instead of performance measure (12-margin pixelwise accuracy) for monitoring early stopping as (1) this subset of size 100 may not truly represent/approximate the distribution of validation set, (2) computing accuracy over all validation set will presumably increase training time. Also, another perspective for the decision is due to mitigate the undesired effects of the model due to short cutting and premature stopping of training, and This risk might be the case due to the dataset of faces and individual observations are close to each other, and color distribution is probably (after quick visual inspection) not close to uniform as most face images represent common skin colors and due to that a flat RGB image with a single or few colors where they're the mode/modes of the population distribution might have high 12-margin accuracy, but also high MSE, and hence will yield to undesired outputs from the model at least for baseline setup search. This can be prevented other methods further, but for baseline search we chose to monitor validation loss over iterations to determine early stopping for the reason mentioned here. Additional details about the hyperparameters and brief approach is given in 3.1.

We also tried a simpler algorithm which tracks and determine early stopping by checking improvement in a relative manner (judged by the ratio of previous and current), but we decided to not use it after few trials though it can be improved, the detail for this algorithm are given in 5.1.

We will also use the name *val100* throughout the paper to refer subset of validation set of 100 randomly sampled observations which is held fixed for all experiments. Also, for visualization purposes we drawn three random samples of size 12 from the validation set, the samples drawn from validation set and are independent from each other, so samples could've been overlapping between 12 sized samples; however, they do not. Size of 12 is chosen because it's easier for visual inspection and also enough to fit in a paper or a column on this report. These three samples of size 12 (36 images in total) are named as CS_i, CS stands for *Comparison Subset*, where i is the index of the sample set. CS_i is held fixed once drawn throughout the experiments and reporting. The x-axes of loss plots *Steps* represents steps where the validation losses are computed (i.e. 3 times per epoch, and it does not represent the number of batches).

The illustration scheme for visual outputs of models that we followed throughout the paper is top-row: input (grayscale), middle-row: model prediction (RGB), bottom-row: target (groundtruth RGB).

Finally, for the architecture search/experiments, unless otherwise specified the set of hyperparameters used (other than the ones mentioned above) are $batch\_size = 16$, $max\_num\_epochs = 100$, $min\_num\_epochs = 5$; and $filter\_size = 3$ and $padding = same$ is used in convolutional layers. *min_num_epochs* is added to prevent early stopping from stopping too early, i.e. early stopping condition will not be checked before model is trained for *min_num_epochs* epochs.

1 Baseline Architecture (30 pts)

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

- Discuss effect of the number of conv layers:
- Discuss effect of the number of kernels(except the last conv layer):
- Discuss effect of the learning rate by choosing three values: a very large one, a very small one and a value of your choice:

Before going into discussions, the approach we took is to search the baseline architecture in a hierarchical way with the same order given in the list above. In other words, our search scheme begins with the experiments for number of convolutional layers (where number of kernels (4) and learning rate (0.01) is fixed), and with the experiment giving the best result for number of convolution layers is fixed and move on to the next search parameter (num. of kernels), and so on. However, it should be noted that with this search scheme we may not find best hyperparameter set (including search for model architecture) as we try only a subset of all possible combinations. Thus, we determined a start point of hypermeter set, and gradually searched for hyperparameters by considering the outputs/results of experiments we already did, and select the next experiment (if we conduct) hierarchically based on previous results.

The results of the experiments are given in the Table 1. Note that the experiments in group D are conducted to answer the question *What if we push the architecture to the edge limits ?* (in terms of architecture and within the constraints), and since it is not in the core of the discussions part we detailed group D in 5.3.

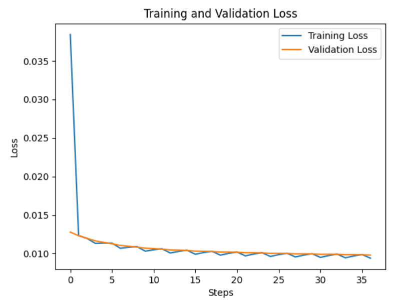
1.1 Effect of number of convolutional layers

The regarding experiment group for this section is group **A**. From the result given in Table 1 the best result with respect to the pixelwise accuracy achieved by *A-2* where the number of kernels is 2, and the number channels and learning rate are fixed to 4 and 5e-2 respectively. This setup reached 14.72% pixelwise accuracy in val100 subset.

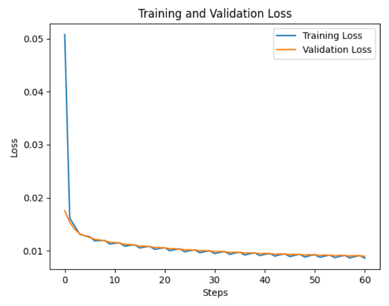
It should be noted that the architecture with only a single convolutional layer, *A-1*, (only output layer without any nonlinear activation function) reached a very similar result in terms of pixelwise accuracy. Furthermore, this setup converged faster compared to *A-2*, however in terms of loss values it's not as good as, but close to *A-2*. There's no non-linearity in *A-1*, which may prevent model from learning useful features, and restricts the model capacity. However, it still learns somewhat okay, and this is probably due to the nature of the task (i.e. image colorization). In other words, mapping from RGB to Grayscale is

Experiment ID	Num. layers	Num. kernels	lr	Number of epochs-batch	train-loss	val-loss	val100 Acc.
A-1	1	—	5e-2	**12-100**	0.0094	0.0098	0.1472
A-2	2	4	5e-2	22-300	**0.0086**	**0.0090**	**0.1472**
A-3	4	4	5e-2	22-200	0.0093	0.0095	0.1466
B-1	2	2	5e-2	22-300	0.0095	0.0094	0.1464
B-2	2	8	5e-2	18-100	0.0087	0.0091	0.1471
C-1	2	4	5e-3	34-100	0.0116	0.0120	0.1470
C-2	2	4	1e-2	63-100	0.0093	0.0097	**0.1476**
C-3	2	4	1e-1	15-100	**0.0085**	**0.0089**	0.1474
C-4	2	8	1e-2	21-100	0.0105	0.0109	0.1471
C-5	2	8	1e-1	15-100	0.0086	0.0090	0.1474

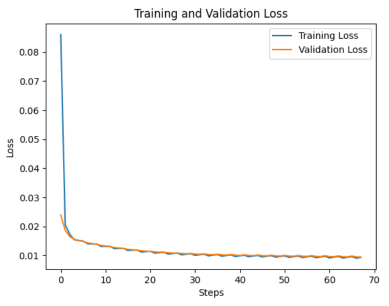
Table 1: Results of the experiments conducted for overall hyperparameter search. *Experiment ID* column shows the hierarchical experiments the next group is determined based on the all experiments preceding that group. We highlighted best results over experiments accounting past experiments up to that experiment (e.g. B-2 does not improve over group A, but C-2 does over A and B).



(a) Experiment A-1: n_layer=1



(b) Experiment A-2: n_layer=2



(c) Experiment A-3: n_layer=4

Figure 1: Loss plots of the experiments in group A.

linear, and the vice-versa can be constructed by a linear map learned over channels although not perfect performance would be expected.

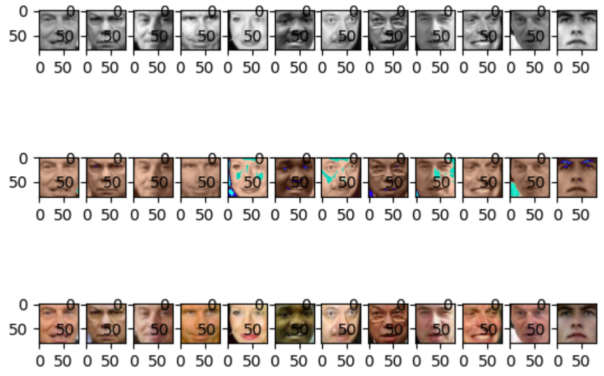
We can argue that 4 layer model might be too complex to learn from the training dataset, and the variety of examples and/or the size may not be large enough to build the internal representation for all (or almost all) different color distribution in face images in the training set.

From the loss plots given in Figure 1. All of the experiments training seems stable with the given setups. We may try to decrease the learning rate for *A-1*, which we will touch here in later sections.

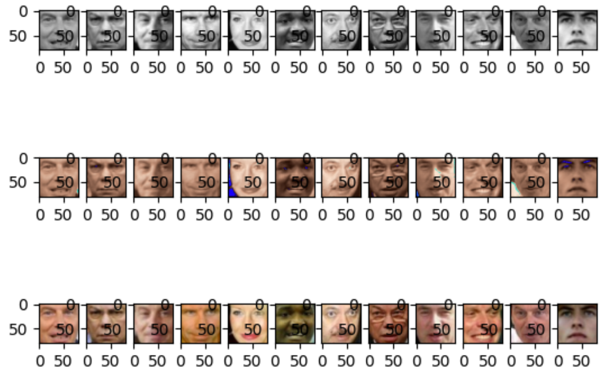
Qualitatively examining results given in Figure 3, *A-2* seems to produce plausible results as parallel to quantitative results. All experiments in group A seem to match the face shape (it's like an identity mapping to each RGB channels), and also the color transitions in general are kind of correctly generated/matched (e.g. shadows, background) spatially.

There are though few problems with A-1 and A-3 as they seem to wrongly interpret some locations with relatively high in color intensity (e.g. illumination effects, white background). They somehow interpret turquois/light-blueish pixels for these regions where A-3 seems to suffers less than A-1 which might be due to non-linear transformations and being a more complex model, whereas A-2 does not seem to suffer from this problem as it generates pixels with matching colors for these regions.

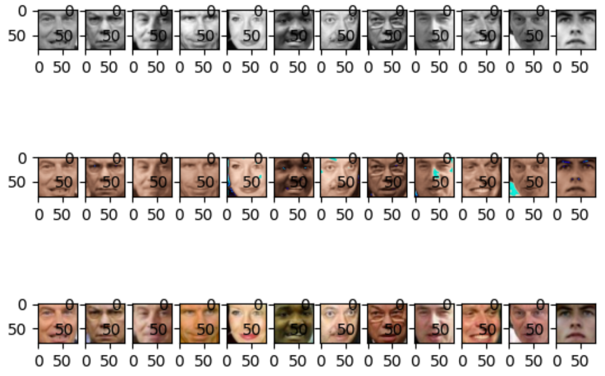
Another common problem apparent in this experiment group (this time A-2 is also included along with



(a) Experiment A-1



(b) Experiment A-2



(c) Experiment A-2

Figure 2: Visualization for experiments in group A on CS_1. Top-row: input (grayscale), middle-row: model prediction (RGB), bottom-row: target (groundtruth RGB).

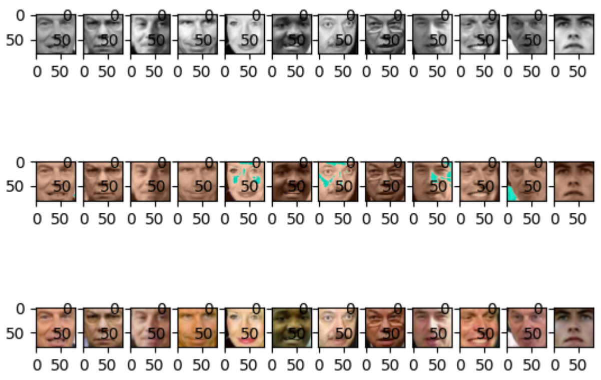
A-1 and A-3) that they wrongly interpreted high contrast regions (e.g. eyebrows, dense black background) with dark-blue/navy pixels. The blueness can be due to initialization, or models somehow find a shortcut (not skip connections) such that these pixels also decrease the loss.

With this examination, we chose to extend A-2 by trying to further improving it in the upcoming search as it performed better than its peers (experiments in the same group).

1.2 Effect of number of kernels

The regarding experiment group for this section is group **B**. Experiments in this group are extended search on setup of A-2. The best result is achieved by B-2 in this group, but no further improvement was seen over A-2 although the quantitative results are very close to A-2.

The visual outputs for group B are given in Figure 2. Comparing Figure 2 along with Figure 2b, it's seen that the low number of kernels (B-1) has led to deficiencies in brighter regions (the problem discussed in 1.1). Similar to outputs discussed in 1.1, these pixels for generated images are turquois/light-blue


(a) Experiment B-1

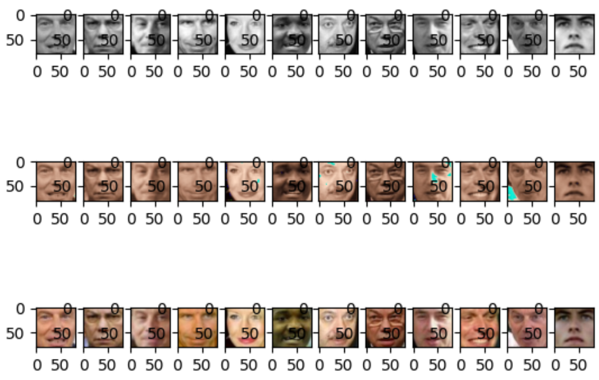

(b) Experiment B-2

Figure 3: Visualization for experiments in group B on CS_1.

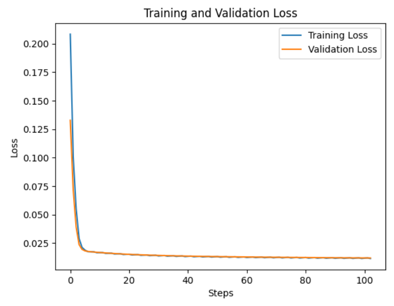
again, but B-1 suffers a lot from this compared to B-2. On the other hand, surprisingly the shortcoming of A-2 discussed in 1.1 that is predicting dark-blue colors for high contrast-low intensity regions are no longer present for both B-1 and B-2. Overall, A-2 still outperforms visually the experiments in group B. Therefore, we move on with experimenting on A-2 in the next search group C.

1.3 Effect of learning rate

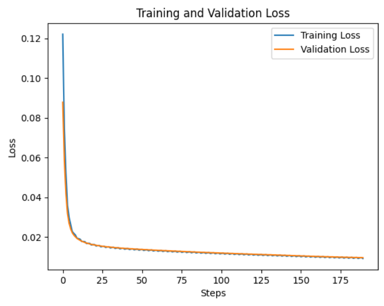
In this experiment group we selected experiments A-2 and we also give B-2 a shot to improve itself for searching learning rate. The experiments C-1, C-2, C-3 are with base setup of A-2, and C-4 and C-5 are based on B-2.

Here, for the experiments C-1, C-2 and C-3 the increase in the learning rate resulted in a positive effect as the best results of group A (spesifically A-2) are improved. The first increase in the learning rate (to 1e-2) C-2 achieves the best val100 accuracy score, but this measure cannot be generalized as val100 consists of 100 randomly sampled data points and there are 2000 examples in the validation set. The increase in learning rate increased the convergence speed as expected without any harm on generalization error for setup of A-2. The loss plots of the experiments regarding change in the learning rate is given in Figure 4. The base setup for experiment group C-1, C-2 and C-3 was A-2, and A-2 is stopped training at 22-300, where C-1 and C-2 surpassed this (slowing convergence) with 34-100 and 63-100 respectively. C-1 couldn't converged to a good point and loss value is relatively higher overall comparing groups A,B,C, so we will not include it in our visual inspections. C-2 so far got the highest accuracy w.r.t val100 subset, but it took almost 3 times more time to converge to that point, also in terms of loss, it couldn't reach to A-2. On the other hand, increasing the learning rate to 0.1 (C-3) helped a lot to this specific setup by converging 7 epochs earlier and also loss values surpassed A-2.

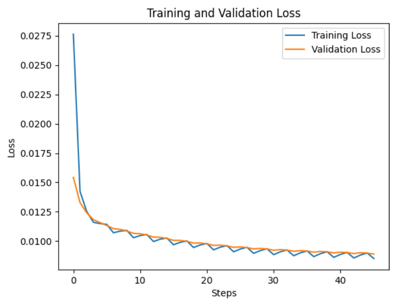
The model predictions for C-2 and C-3 are given in Figure 5 and since the difference from visual comparsion was not too obvious on CS_1 alone, we provided the outputs on CS_2 and CS_3 also in Figure 9. Visually inspecting the outputs of C-2, C-3 along with A-2 (so far the best) overall it can be said that outputs of C-2 is more preferable over A-2 and C-3 although C-2 could not achieve A-2 and C-3 in terms of loss, but it outperformed both A-2 and C-3 on val100 set accuracy. To rep up, we will consider C-2 as our final baseline setup, but we note that we might also try C-3 in upcoming experiments if desired as it might have potential when improving further with additional aids.



(a) Experiment C-1: lr=0.005

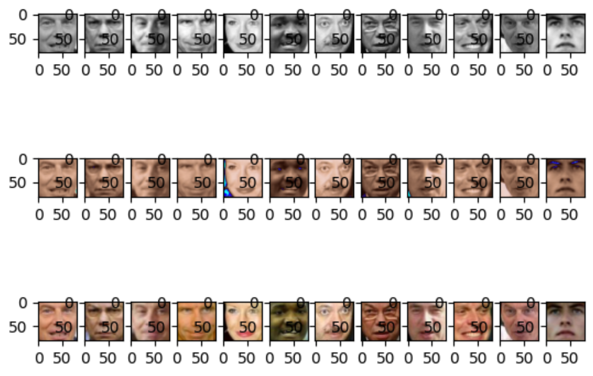


(b) Experiment C-2: lr=0.01

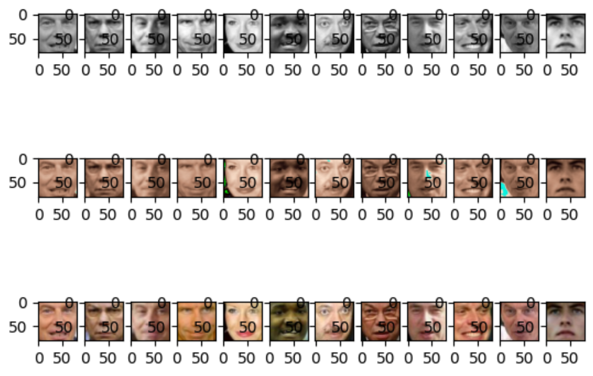


(c) Experiment C-3: lr=0.1

Figure 4: Loss plots of the experiments in group C.



(a) Experiment C-2: lr=0.01



(b) Experiment C-3: lr=0.1

Figure 5: Visualization for experiments C-2 and C-3 on CS_1.

Experiment ID	**Num. layers**	**Num. kernels**	**lr**	**Number of epochs-batch**	**train-loss**	**val-loss**	**val100 Acc.**
C-2	2	4	1e-2	63-100	0.0093	0.0097	**0.1476**
C-3	2	4	1e-1	15-100	**0.0085**	**0.0089**	0.1474
C-2+BN	2	4	1e-2	19-300	0.0138	0.0118	**0.1477**
C-3+BN	2	4	1e-1	12-100	0.0110	0.0087	0.1473
C-2+Tanh	2	4	1e-2	23-100	0.0108	0.0113	**0.1477**
C-3+Tanh	2	4	1e-1	17-100	0.0087	0.0091	0.1476
C-2+Tanh+L16	2	16	1e-2	25-100	0.0101	0.0105	**0.1477**
C-3+Tanh+L16	2	16	1e-1	11-100	0.0088	0.0091	0.1475
C-2+Tanh+L16 +SepConv	2	16	1e-2	26-100	0.0114	0.0119	**0.1481**
C-3+Tanh+L16 +SepConv	2	16	1e-1	18-100	0.0088	0.0091	0.1474

Table 2: Results of the experiments conducted to further improve baseline models. We selected two candidate models C-2 and C-3 for baseline as both of them were comparable in terms of performance and visually plausible predictions.

2 Further Experiments (20 pts)

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

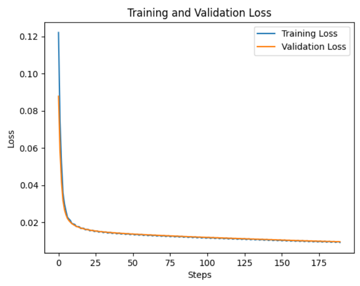
- Try adding a batch-norm layer (torch.nn.BatchNorm2d) into each convolutional layer. How does it affect the results, and, why? Keep it if it is beneficial.
- Try adding a tanh activation function after the very last convolutional layer. How does it affect the results, and, why? Keep it if it is beneficial.
- Try setting the number of channels parameter to 16. How does it affect the results, and, why? Keep it if it is beneficial.
- 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.

As we also mentioned in 1.3 we selected C-2 as our final baseline architecture, and here we will try to further improve this setup. However, as we also mentioned earlier we will give C-3 a shot since quantitatively it is also promising (e.g. in terms of validation loss).

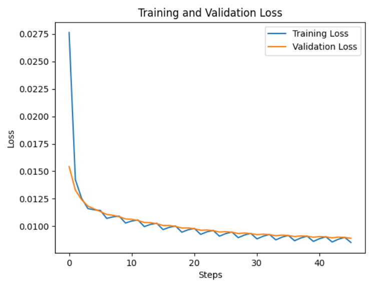
The results of the experiments in this group are given in Table 2. It is important to note that in this section we will comment and make decisions based on mostly quantitative measures and metrics because the model predictions at this point are almost indistinguishable from each other between not only within an experiment group (e.g. C-2+BN vs. C-3+BN), but also between groups (e.g. C-2+BN vs. C-2+Tanh vs. C-2+Tanh+L16). Having said that we will give visual outputs for some experiments in 5.4 for further reading.

2.1 Adding Batch Norm

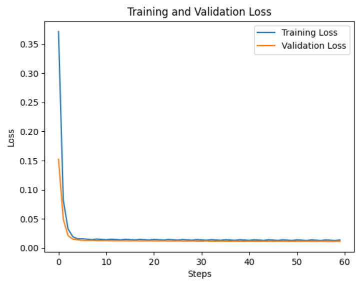
Since the question asks to add B.N to each convolutional layer, we also added BN to last conv. layer. I'm not sure this is intended (usually it's not preferable to use BN in very last layer of network), but the experiment conducted and reported this way.



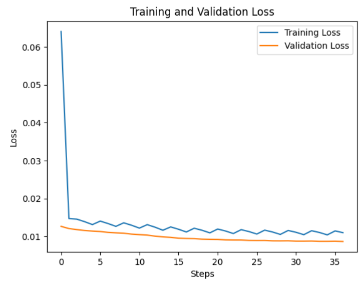
(a) Experiment C-2: lr=0.01



(b) Experiment C-3: lr=0.1



(c) Experiment C-2+BN: lr=0.01



(d) Experiment C-3+BN: lr=0.1

Figure 6: Loss plots of the experiments in group C.

From the loss plots of BN experiment given in Figure 6, BN applied to C-2, kind of helped increasing convergence speed, but it started to plateau too early and couldn't improve too much while for C-3 it hurt performance as it slowed down the training of model of training set.

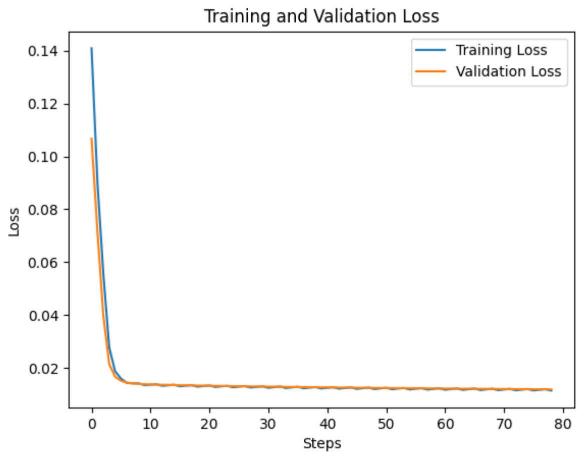
Adding batch norm had a negative effect on models and hurt the performance of both of the baseline networks C-2 and C-3. This might be due to the fact that our baseline models are kind of shallow networks (depth is small) as only 1 hidden conv. layer followed by the output conv. layer.

Another reason of this negative effect might be the batch size as the performance of batch-norm directly affected by the batch-size although intuitively 16 is pretty decent in terms of having a diverse batch during training and the reason is unlikely to be the batch-size. Related to this, diversity of the dataset can also be considered as the diversity of the dataset is relatively low consisting of faces only and adding batch-norm just adds a non-inherent difficulty for model to learn, i.e. since the diversity is low it could make nearly no effect, but it also makes the learning more difficult for model by both normalization and introducing learnable parameters (this is negligible). Furthermore, if the distribution of the training set and the validation set is different that may cause problems because B.N. component usees training set statistics in test/prediction time.

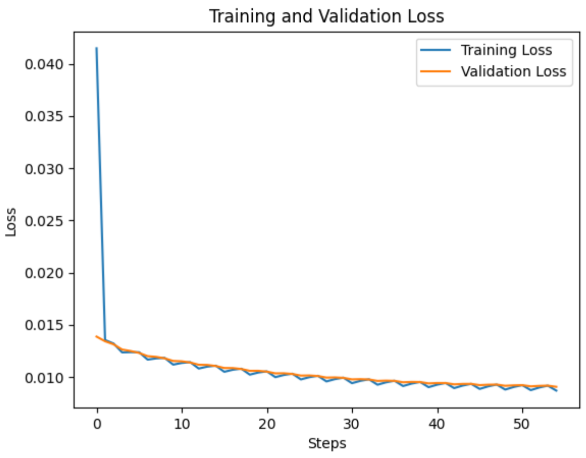
All in all, we decide to not use batch-norm in the upcoming experiments.

2.2 Adding Tanh Activation

Adding Tanh activation seem to help the network to learn the representations easier. This is expected as $tanh(x)$ function maps the inputs to the range $(-1, 1)$ and we normalize the target images to $[-1, 1]$, so Tanh activation function kind of clipping the results of the model to be in range $(-1, 1)$ which alleviates the problem of model learnining to output values in range $[-1, 1]$, it is inherently done with Tanh activation function. We will keep Tanh function as the activation function of the last layer.



(a) Experiment C-2+Tanh+L16+SepConv: lr=0.01



(b) Experiment C-3+Tanh+L16+SepConv: lr=0.1

Figure 7: Loss plots of the final setups for experiments conducted under *Further Experiments*.

2.3 Increase to 16 kernels

Increasing number of kernels to 16 which is effectively increasing number of kernels for the intermediate convolutional layer only as the setup has 1 intermediate layer, seem to help converging to a better parameter set as both training loss and val loss decreased for C-2+Tanh+L16, and it has little to no effect for C-3+Tanh except it helped for faster convergence (almost 6 epochs earlier). We will keep this setup as we set number of kernels to 16 in upcoming experiments.

2.4 Using Depthwise Separable Convolutions

The loss plots regarding the two models are given in 7.

Since our baseline architecture has only 1 intermediate layer and the number of channels in output layer is not divisible by 16 (selected in the previous experiment) we decided to use Deptwise Separable Convolutions [1] (deptwise conv. followed by pointwise conv.) by utilizing grouped convolutions with C_{in} and C_{out} being the number of input channels for that layer (i.e. 1 for first conv. layer) and with groups in that conv. layer also equals to C_{in} followed by pointwise conv. layer with 1x1 kernel to C_{out} (i.e. 16 for first conv. layer). This effectively increases the depth of the model, but decreasing the number of learnable parameters (i.e. in first conv layer (intermediate layer) the number of learnable parameters are $1 * 16 * 3 * 3 = 144$ and in last layer $16 * 3 * 3 * 3 = 432$ which sums up to 576, but with separable convs. first layer $1 * 16 * 3 * 3 + 1 * 16 * 1 * 1 = 160$ and for second (last) layer $1 * 16 * 3 * 3 + 3 * 16 * 1 * 1 = 192$ which sums up to 352).

Deptwise Separable Convolutions does not only helps reducing number of parameters and FLOPs in a efficient way, they also help model to learn channelwise interactions. Moreover, ability to learn channelwise interactions would be expected to help in this particular task (image colorization) as we want model to learn RGB channels for each grayscale input. The stopping point of training is similar to C-2+Tanh+L16 without Sep. Conv., but the loss value is increased a little compared to that experiment while C-2+Tanh+L16+SepConv achieved the highest val100 accuracy so far. It seems that for C-3+Tanh+L16, adding SepConv doesn't affect the model performance in anyway likewise L16 adddition to C-3+Tanh.

After conducting all experiments in this group both to C-2 baseline and C-3 baseline, we will keep separable convolutions. Therefore, our final configuration is C-2+Tanh+L16+SepConv (due to the best val100 accuaracy).

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 plot of the training mean-squared error loss 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:
- Discuss the advantages and disadvantages of the model, based on your qualitative results, and, briefly discuss potential ways to improve the model:

3.1 Automatically chose number of epochs

To choose epoch numbers automatically, we utilized an early stopping algorithm which takes few hyperparameters into account and based on these hyperparameters it automatically choose where to stop training. We detailed this in *Preabmle* as it is the adopted applied this method throughout the paper and for all experiments. Briefly explaining the methodology again, we computed validation loss (over all validation set) three times per epoch and check this loss value during training to determine where to stop during training. Hyperparameters to be chosen in this method, namely *patience* and *delta*, are set by observing few iterations during mock training to see how the model behaves overall and after setting those hyperparameters, they are held fixed throught all experiments we conducted.

The descriptions for hyperparameters of the algorithm are provided in docstrings in the codebase, but a brief overview for patience and delta are;

- **patience**: How many comparison should the algorithm wait (for no-improvements) before making the decision to stop training. Note that if an improvement occurs within counting for the patience, the count is reset (e.g. no-imprv.(count:1), no-imprv.(count:2), imprv.(count:0), and so on).
- **delta**: The threshold (lower bound) for deciding on improvement. In other words, the difference between current and previous value is determined as improvement if it is greater than delta. The algorithm by default sets the approach as lower is better (e.g. loss), and the higher is better (e.g. accuracy) can be tracked easily by giving the value multiplied by -1.

3.2 Loss plot

Loss plot related to our best configuration is given in Figure 7a.

3.3 Accuracy plot

Accuracy plot related to our best configuration is given in Figure 8. From the plot we see a high accuracy at very early stage of training and a sudden dramatic decrease only after 2-3 epochs, and the accuracy then starts to increase slowly by the following iterations. The sharp spike followed by a sudden decrease at the very early stage of training is probably due to the fact that model finding a shortcut, this risk is mentioned in detail in *Preamble* in the discussion of what measure to provide to check early stopping condition.

3.4 Qualitative results

The visual outputs for the best model including input (grayscale), output (predicted RGB) and target (groundtruth RGB) is given in Figure 10.

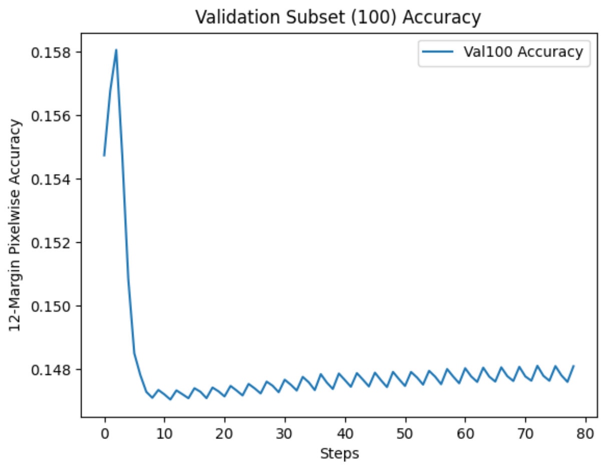


Figure 8: 12-Margin pixel-wise accuracy plot of (best) experiment C-2+Tanh+L16+SepConv measured over *val100*.

3.5 Advantages and disadvantages

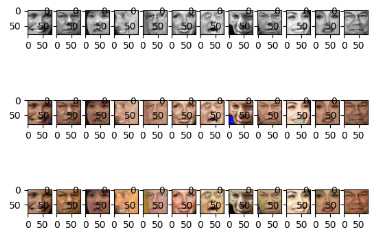
We compare our final configuration (C-2+Tanh+L16+SepConv) (best model in upcoming contexts) with the related baseline configuration (C-2) that is extended.

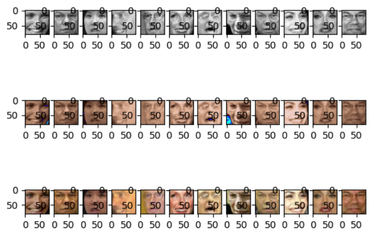
We already mentioned some shortcomings of C-2 in 1 and some of which are also seen in other experiments conducted for baseline configuration search (e.g. C-3, A-2). Some of the shortcomings during baseline search is that some of the predictions have deficiencies/sensitivity to certain cases when visually inspected. These deficiencies include but not limited to wrongly predicted pixels on locations where illumination effects occur, or commonly very bright regions, pixels of these regions predicted as light-blueish/turquois colors. Apart from this, high contrast, dense and pixels in low brightness regions were predicted as dark-blue/navy (rarely dark-green for few models) colors. These shortcomings were alleviated by our best configurations, and there's no visible deficiencies for best model predictions like mentioned in this context. It almost always perfectly predicts the shape, skin-color, good prediction for regions with high illumination effects, etc.

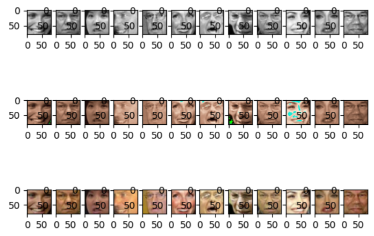
While outputs of our best model is plausible, it's not perfectly predicting colors for all cases, and it can be further improved in this manner. After visual inspection over validation set predictions, there are low amount of flaws/imperfections of the model occurring where the color map is potentially pretty diverse (RGB) for an input (grayscale), like makeups, especially lipsticks with high amount of red pixels are mostly predict as red-brownish colors. Another case is when the input has a relatively high region of background, in this case model mostly outputs background pixels close to skin-colors, this is probably due to having relatively small amount of background including images in the training dataset.

To further aid these problems, data augmentation may be applied to the training set (presumably not rotations or translations, but color related augmentations) which would regularize the model and prevents predicting colors close to skin-colors over all regions (e.g. may improve background or make-up cases.). The model architecture may be change to encoder-decoder style with down-sampling by conv. layers (without same padding) and up-sampling by deconv layers with or without pooling layers. Other activation functions can be experimented. The model complexity can be increased along with adding regularization like Dropout [3]. A good improvement would be (theoretically speaking) to include skip-connections [2] as the task is inherent to learn kind of identity mappings where residual connections would ease the learning.

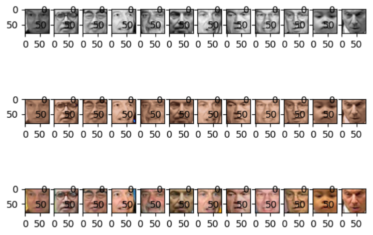
Some of the suggested further improvement experiments require trial-and-error phases, but for some the search space may be reduced by educated guesses or hierarchial experiment scheme.


(a) Experiment A-2: lr=0.05


(b) Experiment C-2: lr=0.01


(c) Experiment C-3: lr=0.1

(d) Experiment A-2: lr=0.05


(e) Experiment C-2: lr=0.01

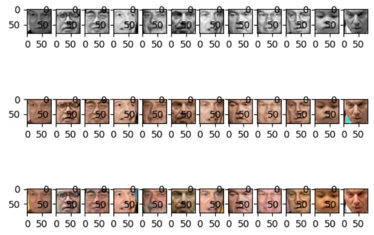

(f) Experiment C-3: lr=0.1

Figure 9: Model predictions on CS_2 and CS_3 for A-2, C-2, C-3.

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:

The `template.py` is ready to use for such a problem, all you have to do is to run this script. This script will first train the network (with the exact same seed), and then writes the serialized numpy file for test set, assuming test dataset is located under `<PROJECT_ROOT>/ceng483-f22-hw3-dataset/test`. Once the run is complete, you should be able to see `test_estimations.npy` in `checkpoints/<experiment_name>`.

5 Additional Comments and References

5.1 Legacy Early Stopping

We also implemented a simple early stopping algorithm where the algorithm signals stop if there's no improvement above 1% between two consecutive epochs w.r.t validation dataset measured by the loss function (MSE) (i.e. $1 - \frac{L_{current}}{L_{prev}} < 0.01$). One different thing is that this algorithm checks for relative improvement rather than monitoring the difference, the motivation for this is to determine tolerance threshold more inherently, and also it's somewhat adaptive/dynamic as the 1% definition alters the effective threshold for every pair of loss values. Since we did not use this algorithm in any experiment reported in the paper, we will skip the detailed explanation. However, the implementation detail can be found in the codebase.

5.2 Experiments in Group C

As the visual outputs are not easily distinguishable, and there are some subtle differences that are hard to spot, we additionally provided model predictions on CS_2 and CS_3 in Figure 9.

Experiment ID	**Num. layers**	**Num. kernels**	**lr**	**Number of epochs-batch**	**train-loss**	**val-loss**	**val100 Acc.**
D-1	1	—	1e-2	24-100	0.01060	0.0110	0.1466
D-2	1	—	2.5e-2	17-100	0.0101	0.0105	0.1467
D-3	1	—	5e-3	21-100	0.0108	0.1123	0.1466
D-4	4	8	1e-2	33-100	0.0103	0.0107	0.1468
D-5	4	8	5e-3	45-100	0.0110	0.0115	0.1470

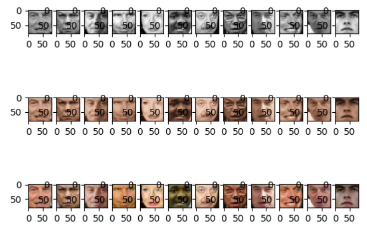
Table 3: Results of the experiments conducted for group D.

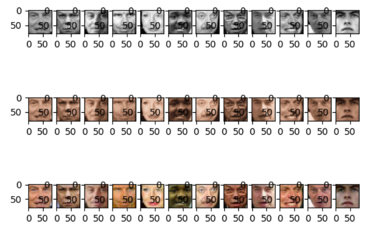
5.3 Experiments in Group D

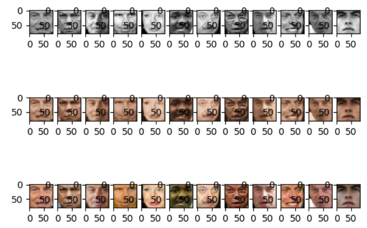
In this section we provided the results for experiments whose configurations is through the end of the constraints (e.g. 0.1 (max lr) and 4-8 (max num layer-num filters), etc). The results are provided in Table 3 for further readers.

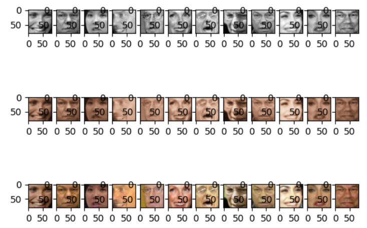
5.4 Visualization for Further Experiments

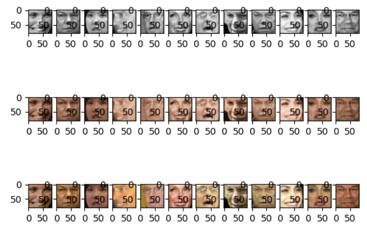
In 2 we mentioned that visually outputs of the models are indistinguishable both within and between the experiment groups. Visual outputs for some models are given in Figure 10 and 11 for C-2 and C-3 baseline extensions respectively.

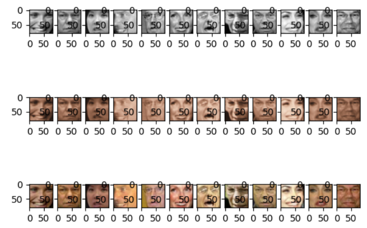

(a) Experiment C-2+Tanh: lr=0.01


(b) Experiment C-2+Tanh+L16: lr=0.01


(c) Experiment C-2+Tanh+L16 +SepConv: lr=0.01


(d) Experiment C-2+Tanh: lr=0.01


(e) Experiment C-2+Tanh+L16: lr=0.01

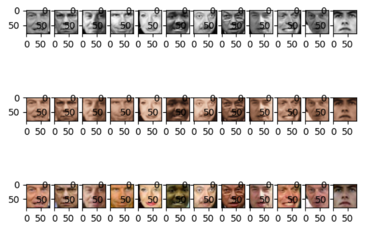

(f) Experiment C-2+Tanh+L16 +SepConv: lr=0.01

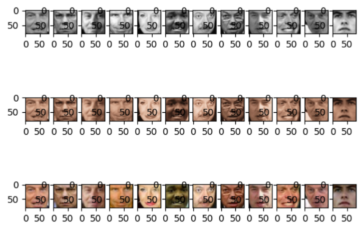
(g) Experiment C-2+Tanh: lr=0.01

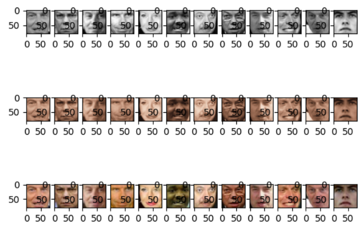
(h) Experiment C-2+Tanh+L16: lr=0.01

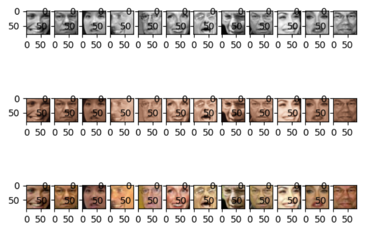
(i) Experiment C-2+Tanh+L16 +SepConv: lr=0.01

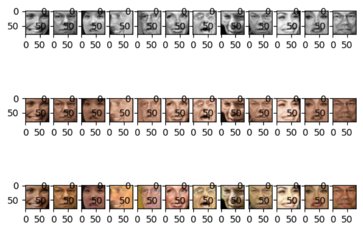
Figure 10: Model predictions on CS_1, CS_2 and CS_3 for C-2+Tanh, C-2+L16 and C-2+Tanh+L16+SepConv.

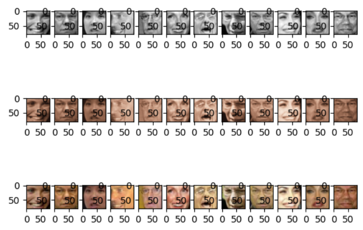

(a) Experiment C-3+Tanh: lr=0.1

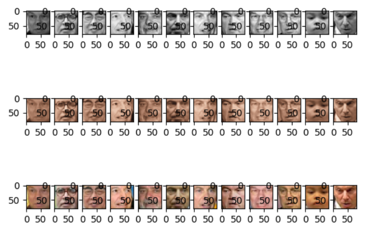

(b) Experiment C-3+Tanh+L16: lr=0.1

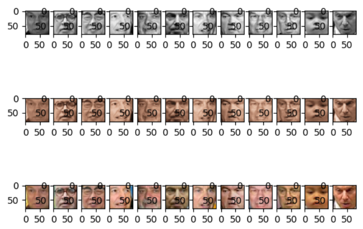

(c) Experiment C-3+Tanh+L16 +SepConv: lr=0.1


(d) Experiment C-3+Tanh: lr=0.1


(e) Experiment C-3+Tanh+L16: lr=0.1


(f) Experiment C-3+Tanh+L16 +SepConv: lr=0.01


(g) Experiment C-3+Tanh: lr=0.1


(h) Experiment C-3+Tanh+L16: lr=0.1

(i) Experiment C-3+Tanh+L16 +SepConv: lr=0.1

Figure 11: Model predictions on CS_1, CS_2 and CS_3 for C-3+Tanh, C-3+L16 and C-3+Tanh+L16+SepConv.

References

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[4] Bjarte Mehus Sunde. Early stopping for pytorch. `https://github.com/Bjarten/early-stopping-pytorch`, 2018. Accessed: 2023-01-10.