University of Porto - Faculty of Engineering

Computer Vision Report

Image Colorization Analysis

André Santos
up201906573@edu.fe.up.pt
João Andrade
up201905589@edu.fe.up.pt
Maria Carneiro
up201907726@edu.fe.up.pt

May 2023

Contents

List of Figures

1 Introduction

Image Colorization is the process of determining color of an initial grayscale image realistically and it has numerous applications in various fields, such as in medical imaging or entertainment and media. In this report, we explore, analyze and evaluate models that perform image colorization, as well as study its value as a pretext task for Self-Supervised Learning (SSL).

2 Dataset

In order to simplify the training process, we decided to train and evaluate the explored models using the CIFAR10 [1] dataset. It consists of 60000 32x32 colour images in 10 classes, with 6000 images per class, 50000 of which are training data and 10000 are test data. In order to perform validation of the models, we split the training data randomly and gathered 5000 images for our validation dataset [2]. This allowed us to interpret the behaviour of the models during the training process.

3 Task 1 - Image Colorization as a Regression Problem

This task's goal was to design, implement, train and evaluate a model that performed image colorization. Since a lot of models are already proposed in the field, we decided to analyze some of them and test their performance with our dataset, while considering some of alterations of our own.

We started by exploring a ResNet based model [3], proposed by Luke Melas, considering two different loss functions: Mean-Squared Error (MSE) and Learned Perceptual Image Patch (LPIPS). Based on that model, we decided to explore it's implementation making use of a DenseNet. Finally, we analyzed Zhang's colorization model [4] presented at the European Conference on Computer Vision (ECCV), in 2016, in order to test a model not based on pre-trained nets.

For all models, we used a fixed learning rate of 1e-3 and a batch size of 64, to accommodate for GPU memory constraints, and we also used the Adam optimizer. Every model was trained on 50 epochs.

The main structure of the code was inspired by the class learning materials (notebooks) and the colorization notebook [5] developed by SmartGeometry@UCL.

All the results and corresponding graphs referenced in the sections below will be in the appendix.

3.1 Evaluation Metrics

To evaluate our models, alongside the Loss, we also computed the Peak Signal-to-Noise Ratio (PSNR) and the Structural Similarity Index (SSIM). Both these metrics are widely used to evaluate image colorization, since they, generally, measure how close the predicted image resembles its original colorized version: PSNR evaluates the difference between the original and the distorted image and SSIM takes into consideration the difference in luminance, contrast, and structure between the original image and the colorized one. SSIM values represent good similarity when they are near 1 and PSNR when they fall between 30dB to 50dB. Moreover, in comparison with other metrics, they showed better results and higher correlation with subjective evaluation. [6] Subjective evaluation, even if not quantifiable, was another metric we employed to analyze the results: we visually compared the predicted colorized images with their grayscale and original versions, to access if the prediction was confident or not.

3.2 Loss Functions

Defining the loss function has a crucial role in image colorization since it is by its definition that the model can learn to optimize its colorization process to produce confident and realistic results. For all models presented, and since we are covering regression, we use the Mean-Squared Error (MSE), which aims to minimize the distance between the predicted color and the real one. For the first model (ResNet), we also tested a perceptual based loss function called Learned Perceptual Image Patch (LPIPS)[7]. LPIPS is used to measure the similarity between two images considering their human perception and visual representation. We used a VGG based LPIPS loss function to test if a perceptual approach was favorable to an error-based one.[8] However, the multi-modality of the problem can't generally be well handled by loss functions standardly used in regression problems, so dullness is to be expected, since no color re-balancing is performed. That is an alternative to improve our results and obtain more vibrant and saturated colors in the images [4].

3.3 ResNet Model

The ResNet Model is a convolutional neural network that uses the ResNet-18 network as its encoder, to extract features from the images being passed through the model. It fulfills the job of the encoder section of the CNN. Then, 5 deconvolutional layers are applied to upscale the features and return the colorized output. Since ResNet18 has around 11 million trainable parameters and was trained on an extensive amount of data, it is expected that it may outperform other networks built from scratch and trained on our smaller dataset.

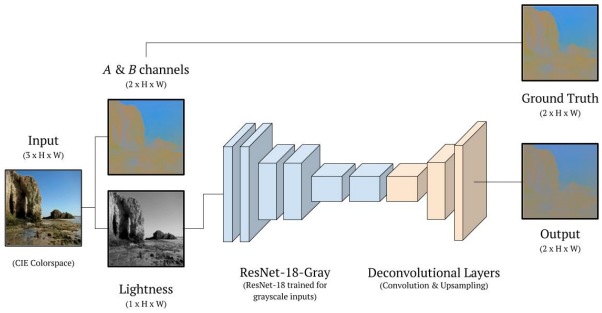
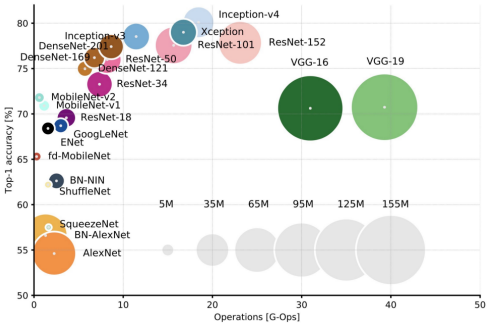


Figure 1: ResNet Based Colorization Model

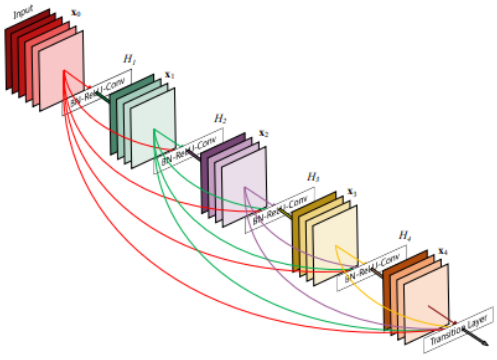
Considering the results in general, using either of the considered loss functions (MSE and LPIPS), the model produced dessaturated images that could correctly predict colors in the blue/green spectrum more easily than to distinguishing bright colors in the original images. However, LPIPS had slightly better performance than MSE, but the differences were minimal and so, negligible. Since LPIPS required more computational performance than MSE, we decided to test the other models considering this loss function.

3.4 DenseNet Model

After evaluating the ResNet results, there seemed to be an opportunity for improvement. Adding extra layers or tuning hyper-parameters started to show less relevance to the results, so searching for a different CNN architecture seemed to be the way to go. While going through information from past lectures, we noticed DenseNets were highly rated in the accuracy/number of operations graph. This is also corroborated with the work on "ResNet or DenseNet? Introducing Dense Shortcuts to ResNet"[9] So we decided to give it a try and apply it to the problem.



(a) CNN Architectures compared using the Accuracy and the number of operations. Taken from the curricular unit slides



(b) DenseNet Architecture

Figure 2: DenseNet

A DenseNet is composed of dense blocks, which densely connect each layer to every other layer within the block, promoting information flow and gradient propagation. Inside each of these, every layer is connected to every other layer. No changes in size are allowed inside the block. The output feature maps of each layer are concatenated with the feature maps from all preceding layers. Transition layers are used to control spatial dimensions and facilitate efficient learning. [10]

In the end the results fell short of the expectations. Initially it was foreseen that using this model would drastically change the results, however, the image colorization and the comparison between the used metrics

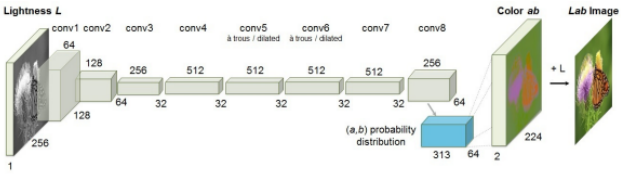
show basically no difference. Despite everything, it was still interesting to study how a different model would behave in this context.

3.5 ECCV Model

Besides the previously mentioned models, we searched on other implementations and investigation work. One of the names that most frequently appeared in various papers was Richard Zhang. Using his paper on Colorful Image Colorization as basis, we tried to mirror the architecture and apply it on our task. [4]

The colorization task architecture is based on a modified version of the VGG-16 network. It is a popular network pre-trained on a large color image dataset for image classification tasks. It is trained in a supervised manner using MSE between the generated colorized images and ground truth color images. After the supervised training, the generator(produces colorized images) and discriminator(classifies images as real or generated) networks are iteratively minimizing and maximizing each other until the first produces visually convincing colorized images.

The output tensor of this convolutional layer has 313 channels, where each channel corresponds to a particular color or color combination from the lab predefined color palette. [11]



(a) Zhang ECCV Model Architecture

Evidently, we could not reproduce the training details mentioned on his work. Besides that, the training data size used on the original architecture is on a totally different scale, so with the available time, data and resources, it wouldn't be possible to achieve the same results. Considering all of that, we experimented training the CNN using the same layer design and looked forward to see where it would go. The results didn't stray far from the previous results, but then again, they did not exceed any possible expectations.

3.6 STL-10 Dataset

We speculated about how the lack of saturation in the output images could be a result of the lack of information in each pictures, given the 32x32 pixels. After the search, we came to the STL-10 dataset, whose image size is 96x96, which increases significantly the information inside each picture.[12] We applied the ResNet model to it, but the results were similar to the CIFAR-10 ones. It could have been because of the low overall length(13000 entries), but we found no correlation in this case.

4 Task 2 - Image Colorization as a Pre-Training (or Pretext) Task

In this task the architecture of the image classifier we construct is intended to use the encoder from the colorization job as the foundation for feature extraction. By utilizing the colorization model's learnt features, we hope to improve picture classification performance by using the encoder's intrinsic representations. In addition, we add layers to the architecture to allow the model to fulfill the categorization task successfully.

The major goal of this task is to compare two unique image classifier model training methodologies. In the first one, we randomly initialize all of the model's parameters, simulating a standard training setup. Second, we use the colorization job as a pre-training phase to establish the image classifier's backbone parameters. By evaluating these two techniques, we hope to assess the effect of colorization pre-training on image classification model performance and convergence.

4.1 Classifiers

The two classifiers we used are implemented using a backbone network, which is a pre-trained model originally designed for colorization tasks. The backbone network extracts high-level features from the input images, which are then used for classification.

4.1.1 Pre-Trained Classifier

In this first classifier we assume that the backbone network has been pre-trained on one of the colorization tasks explained above (Resnet-18).

This way the parameters are not explicitly initialized but rather rely on the backbone model passed as argument. Therefore, this classifier uses the parameters initialized by the pre-trained backbone model which are then used by the classification layer which is added on top of the backbone.

4.1.2 Random Classifier

In this second classifier we also assume the that the backbone network has been pre-trained on one of the colorization tasks explained above (Resnet-18).

But instead of using the parameters that come from the backbone model, the parameters are initialized by iterating over all the modules in the model and using the function $nn.init.xavier_uniform_(module.weight.data)$ to calculate the weights. This function applies Xavier uniform initialization to the modules weight data and sets the weights to random numbers from a uniform distribution. However it keeps the scale of the gradients and activations roughly the same throughout the network, which helps in avoiding the vanishing or exploding gradients problem during training.

Lastly it also sets the model's bias to zero if it exists.

4.2 Training Results

Since the Random Classifier randomizes its parameters, the model has no prior knowledge about the dataset so during training the model learns to recognize patterns and features in the images from scratch.

On the other hand, the Pre-Trained Classifier uses the backbone and has its parameters already initialized with meaningful values. This way, due to the initialization from pretrained weights, the model starts with some prior knowledge about the image features from the dataset and can potentially benefit from transfer learning.

(a) Training Accuracy Comparison

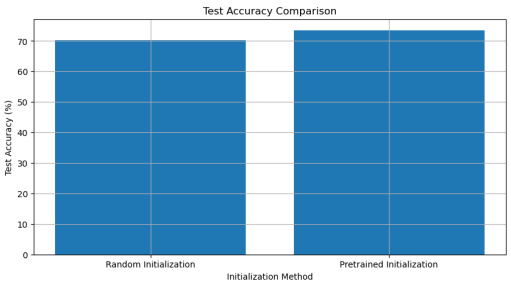
(b) Training Loss Comparison

Figure 4: Metrics for training

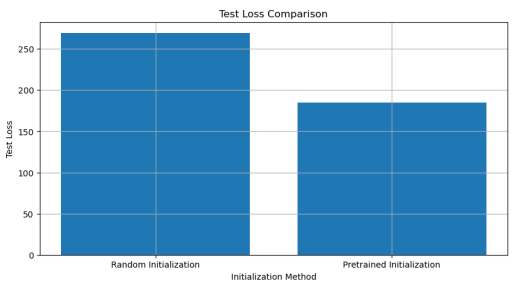
As we can see in the results the training accuracy from the Random Classifier starts much lower than the Pre-Trained one on the first epochs which confirms the previous theory. As we reach the last epochs the model starts to learn and gets closer to the the Pre-Trained values. This also translates to the training loss. In the beginning the loss from the Random Classifier is high because the initial predictions are likely to be far from the correct labels.

4.3 Evaluation Results

Even though the difference was not that big, in the end, the pre-trained classifier ended up with a higher test accuracy and a lower test loss result, which was expected.



(a) Test Accuracy Comparison



(b) Test Loss Comparison

Figure 5: Metrics for test

5 Conclusions

All in all, the results achieved were positive considering the models overall simplicity and the lack of data. The generated images quality depended a lot on the input. We noticed that the most common features, such as the sea and sky blues or the green from vegetation, were frequently well colorized, achieving a near perfect saturation. Cases where the original image was near black and white, the models also performed well. However, some other features were still too close to the gray scale. Besides, there were some objects, cars, for example, who were colorized but kept being painted with a different color than the original. This is understandable, since the training data provides many possible alternatives.

5.1 Results

Generally, the results were similar between all models. The SSIM had low values throughout the models, which makes sense considering it also evaluates both luminance and contrast to compute similarities, and since the images are mostly dessaturated, those contrasts are blurry. That also could be due to the smaller resolution of the images. For the PSNR, the values were satisfactory, being a bit under the threshold of 30dB, for same reasons as the for the SSIM.

Model	Test Loss	Test SSIM	Test PSNR
ResNet MSE	147.256	0.227	21.212
ResNet LPIPS	0.095	0.207	21.088
DenseNet MSE	149.831	0.225	21.190
ECCV MSE	150.343	0.201	21.130

Table 1: Results of metrics applied to all the models.

5.2 Future work

In order to improve the outputs, various work could be done in the future:

- Training the model with a greater dataset, both in size and image dimensions. Extending the training time could help as well, despite the over-fitting possibility.
- Developing or using models significantly more refined than the current ones. Every CNN model with sufficient quality to be used in real-life has significantly more work behind it.
- Use loss functions that are more adapted to the multimodal nature of the problem, considering also color re-balancing to obtain brighter and more saturated images.

References

[1] A. Krizhevsky, "Cifar-10 and cifar-100 datasets." URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[2] S. Alves, "Deep learning in pytorch with cifar-10 dataset," Jun 2020. https://medium.com/@sergioalves94/deep-learning-in-pytorch-with-cifar-10-dataset-858b504a6b54.

[3] L. Melas, "Image colorization with convolutional neural networks," May 2018. https://lukemelas.github.io/image-colorization.html.

[4] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," 2016. https://arxiv.org/pdf/1603.08511.pdf.

[5] S. G. P. Group, "Image colorization." https://colab.research.google.com/github/smartgeometry-ucl/dl4g/blob/master/colorization.ipynb.

[6] I. Žeger, N. Bilanović, G. Šišul, and S. Grgić, "Comparison of metrics for colorized image quality evaluation," in *2022 International Symposium ELMAR*, pp. 209–214, 2022.

[7] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018. https://arxiv.org/pdf/1801.03924.pdf.

[8] C. Ballester, A. Bugeau, H. Carrillo, M. Clément, R. Giraud, L. Raad, and P. Vitoria, "Analysis of different losses for deep learning image colorization," 2022. https://arxiv.org/pdf/2204.02980.pdf.

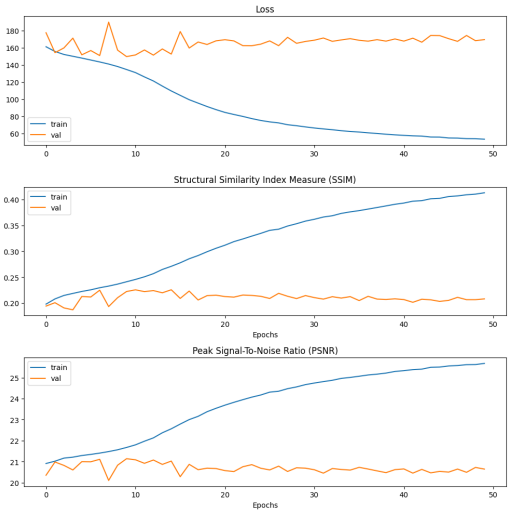
[9] Chaoning Zhang, Philipp Benz, Dawit Mureja Argaw, Seokju Lee, Junsik Kim, Francois Rameau, Jean-Charles Bazin, In So Kweon, "Resnet or densenet? introducing dense shortcuts to resnet," https://arxiv.org/pdf/2010.12496.pdf, Last Access Date: 28/05/2023.

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely connected convolutional networks," https://arxiv.org/pdf/1608.06993v5.pdf, Last Access Date: 28/05/2023.

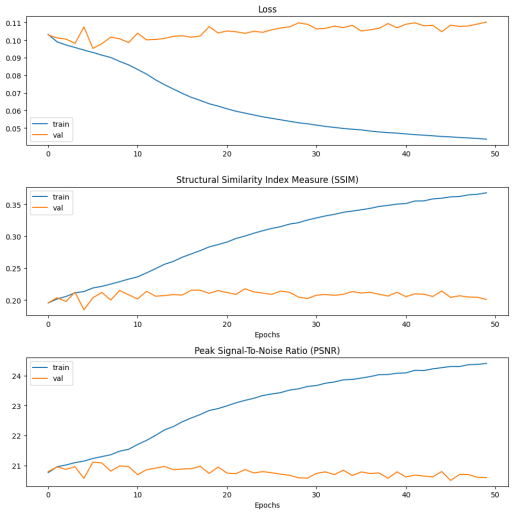
[11] Nidhal K. EL Abbadi, Eman Saleem, "Automatic gray images colorization based on lab color space," https://pdfs.semanticscholar.org/b75c/36245c42d6ad1376dfdcc16f4de6cf7bae10.pdf, Last Access Date: 28/05/2023.

[12] "Stl-10 dataset," https://pytorch.org/vision/stable/generated/torchvision.datasets.STL10.html#torchvision.datasets.STL10, Last Access Date: 28/05/2023.

A Graphs for RESNET Model



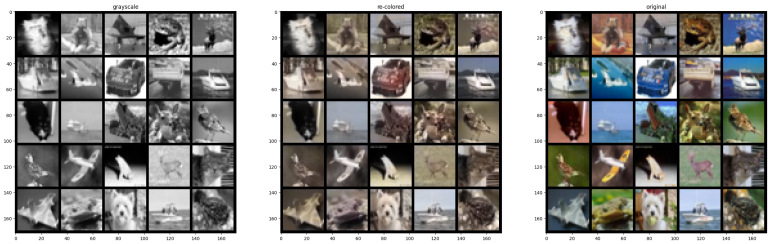
(a) MSE Loss Function



(b) LPIPS Loss Function

Figure 6: Loss, SSIM and PSNR for both cases

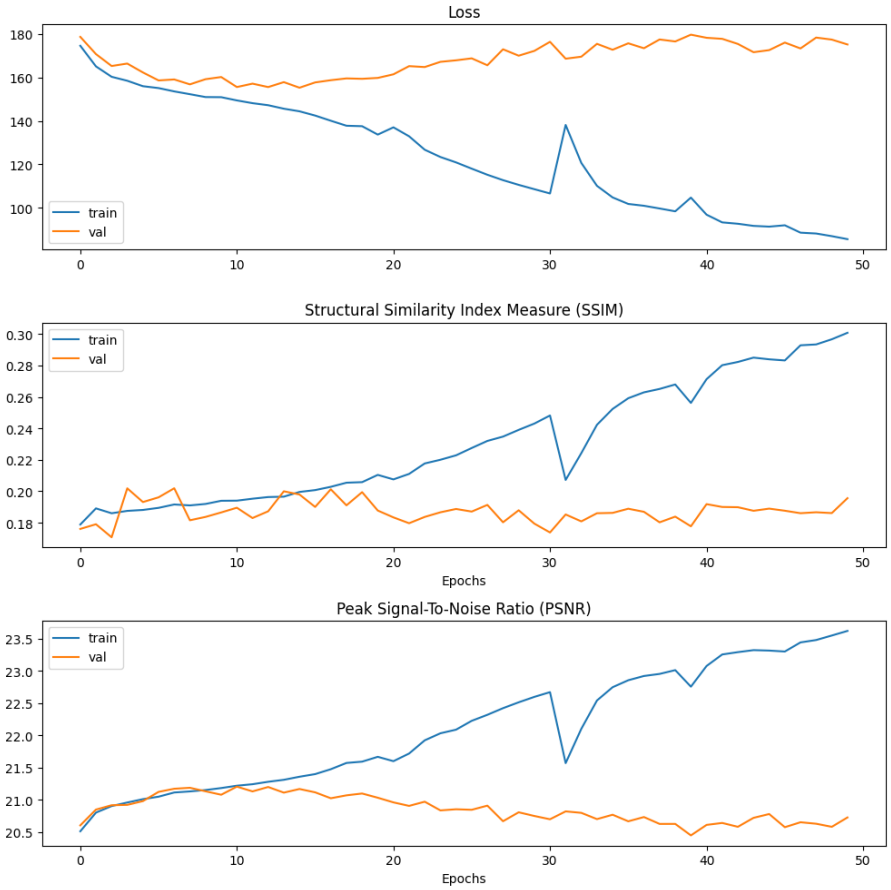
(a) MSE Colorization Results



(b) LPIPS Colorization Results

Figure 7: Colorization Results for both cases

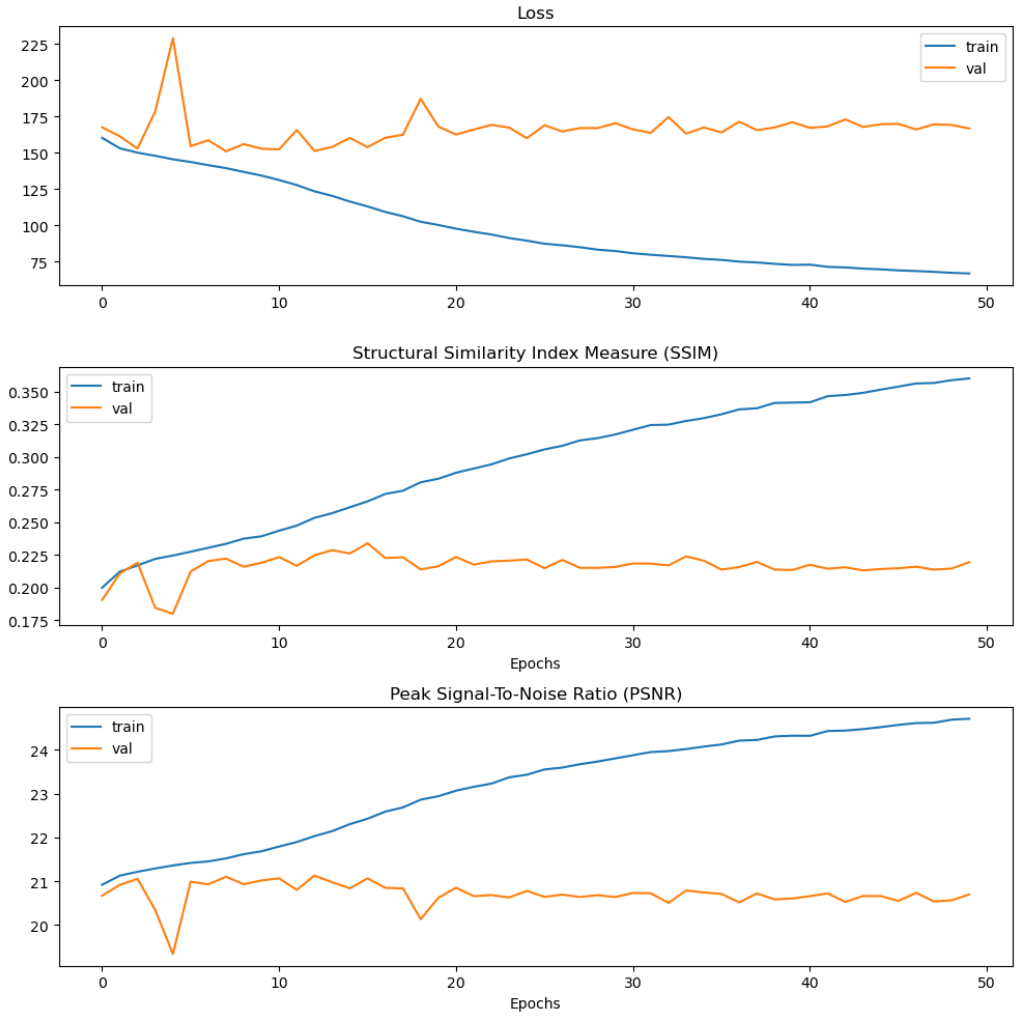
B Graphs for ECCV Model



(a) MSE Loss Function

(a) MSE Colorization Results

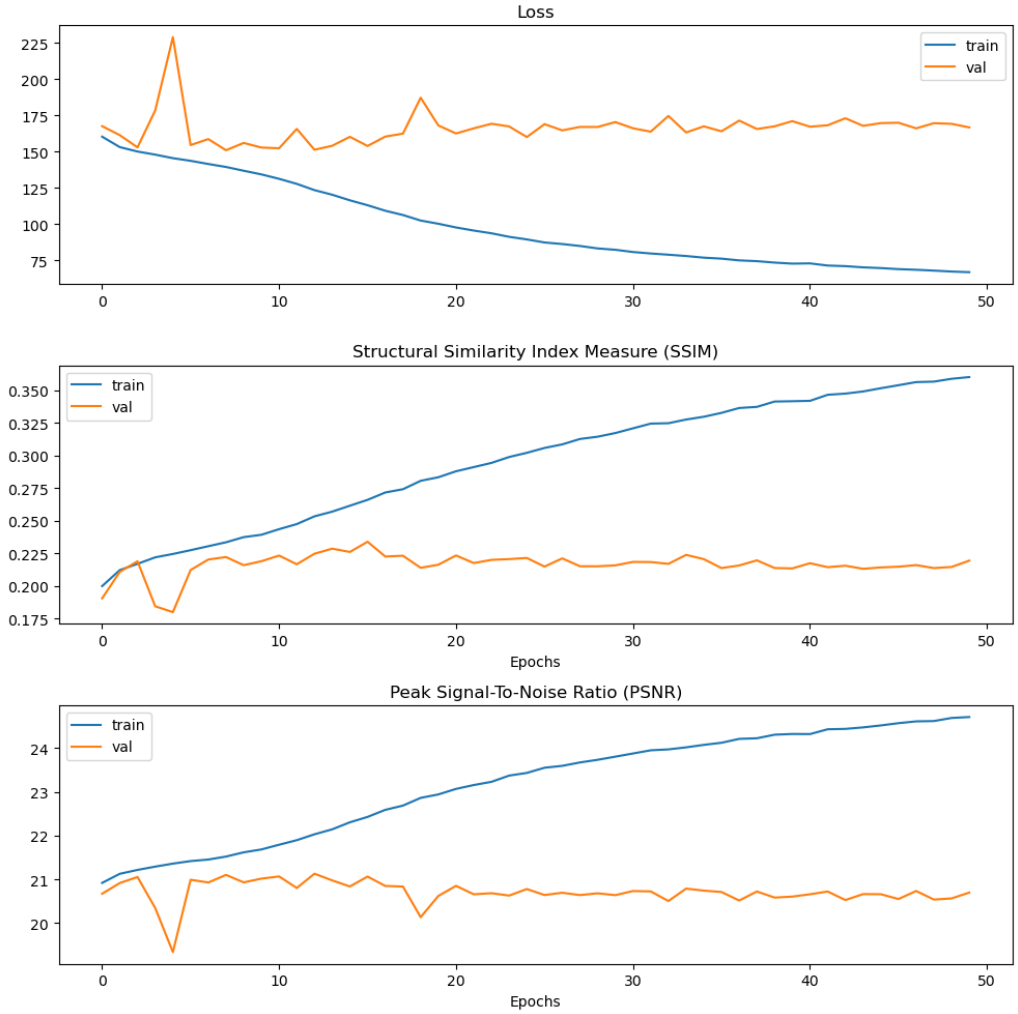
C Graphs for DenseNet Model



(a) MSE Loss Function

(a) MSE Colorization Results

D STL-10 Dataset with ResNet



(a) MSE Loss Function

(a) MSE Colorization Results