APS360: AI Fundamentals

Image Colourization Project

Final Report

Group 10

Word Count: 2483

Members:
Jay Bihola
Raman Mangla
Japvinit Kour
Anikeith Bankar

Table of Contents

Introduction

Whether it is for aesthetics or for understanding the past or to recreate old memories, image colourization has fascinated people since the dawn of neural networks. It has been an important application of artificial intelligence. Our project aims to develop an image processing application to perform such colourization of grayscale images of natural scenes and landscapes. Implementing AI automates the intensive labour-process of colourization, which has historically been done using tools like Photoshop. Additionally, machine learning is the ideal tool for a project such as this one because the colourization process involves an analysis of spatial and colour patterns and sometimes even real-world geographical and historical context to predict the colours.

This project is exciting because ColourizeIt! is aiming to colour historical grayscale images from the 20th century, which would, in turn, provide information about the past. Also, the project has many opportunities for expansion, which makes this project informative for many other fields.

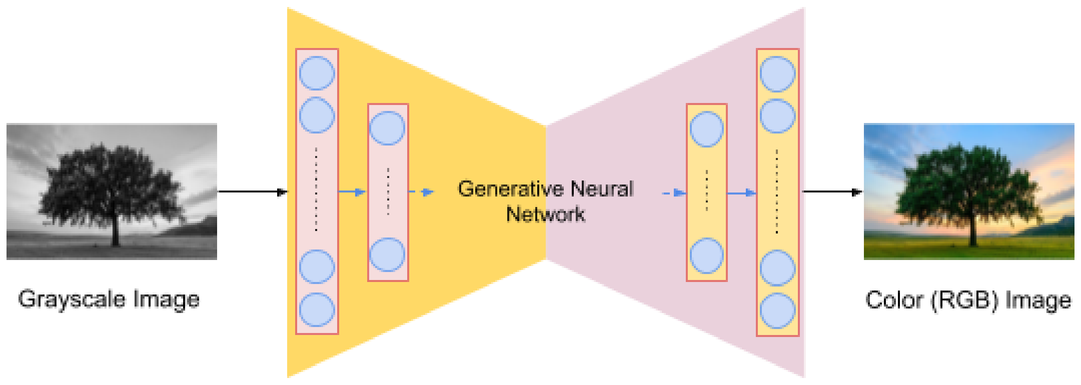


Figure 1: *Overview illustration of the project.*
Source (Tree Image) [1]

Background & Related Work

A lot of work has been done in the field of image colourization using different neural network approaches. One such tool is Algorithmia, which uses cloud-hosted deep learning models [2]. Previous approaches of recolouring images have either required user interaction or produced desaturated colourizations [3]. Algorithmia claims to provide vibrant colours and an automatic approach [3]. The Algorithmia model is based on the model created by *Zhang et al.* [3], which uses a deep learning Convolutional Neural Network (CNN) trained on millions of images from the ImageNet dataset. Because the model has been trained on a wide variety of samples, it produces saturated results when given slightly different images.

Another prominent example is Colourise.sg [4], which is based on Jason Antic's DeOldify [5] Image Colorization project. It uses a deep learning Generative Adversarial Network based model trained on 500,000 old public images from Singapore. It was developed by the Government Technology Agency of Singapore engineers,to colourize old Singaporean photos, due to the lack of results using the Algorithmia tool on images specific to the Singaporean historical context [6]. This tool is able to create high-quality results with believable colour, especially when given images features human subjects and scenery [6].

Data Processing

Since the aim is to find some meaning behind historical images by coloursing them, in order to train the model, various landscape classes were used (as specified in Table 2) as they more commonly show up in historical images.

Images were retrieved from different sources and were of varying sizes. Before the images could be used, they all had to be resized to 144 by 144. In addition, to train the model the images had to be processed to grayscale. A ratio of 80:10:10 was used to split the images into training, validating and testing sets.

Table 1: Sources of the Datasets

Dataset Source	**# of Images**
SiftFlow (Uni. North Carolina) [7]	2690
Landscapes (Kaggle) [8]	4250
Flickr Dataset (GitHub) [9]	3796
LMSun Dataset (Univ. North Carolina) [7]	4005
Urban Setting (Univ. Technology, Sydney) [10]	2838
Total Images	**17 579**

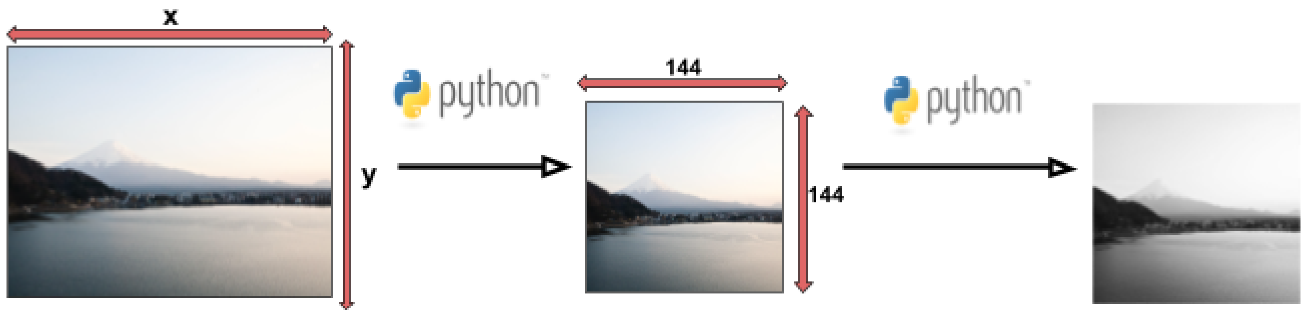


Figure 2: *Input images of varying size resized and processed to grayscale for training.*

Table 2: Statistics of our Datasets

Class Name	Distribution Between Sources	Description of Class
Coasts	SiftFlow: 360 Landscapes: 1050 Flickr Dataset: 1000	Images of land near shore and water bodies; beaches, seas, oceans etc.
Forests	SiftFlow: 320 Landscapes: 800 Flickr Dataset: 681	Images consisting of trees, parks, lots of greenery images
Mountains	SiftFlow: 320 Landscapes: 1200 Flickr Dataset: 678	Images of hills, valleys and mountains
Open Country	SiftFlow: 411 Landscapes: 1000 Flickr Dataset: 492	Images of countrysides, farms, fields, prairies etc.
Streets and Urban Areas	SiftFlow: 608 Landscapes: 150 LMSun: 2468 Urban Setting: 2000	Images of mainly streets/roads/alleyways with some buildings, urban areas
Tall Buildings	SiftFlow: 356 Landscapes: 50 LMSun: 1537 Urban Setting: 838	Images of buildings (front/side views), structures

Architecture

The final model was a Conditional Generative Adversarial Network (cGAN). A GAN consists of 2 adversarial networks, Generator and Discriminator, which end up training each other in the process of improving themselves. The Generator tries to colour the image while the Discriminator looks for incoherence in the Generator's output to label it as a fake. The grayscale image is given as the condition to the model as the generated image needs to be based on it. The Generator and Discriminator architectures were inspired by the Pix2Pix GAN, which was developed by Phillip Isola et al. [11].

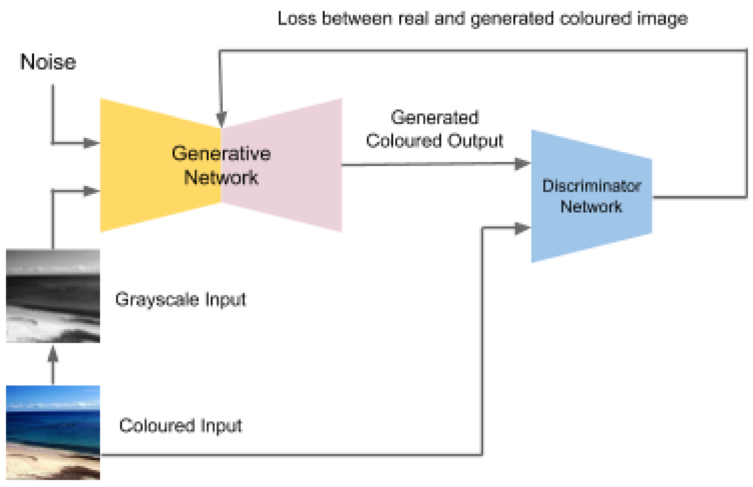


Figure 3: *A Conditional DCGAN training loop for colourizing grayscale images.*

Generator Network

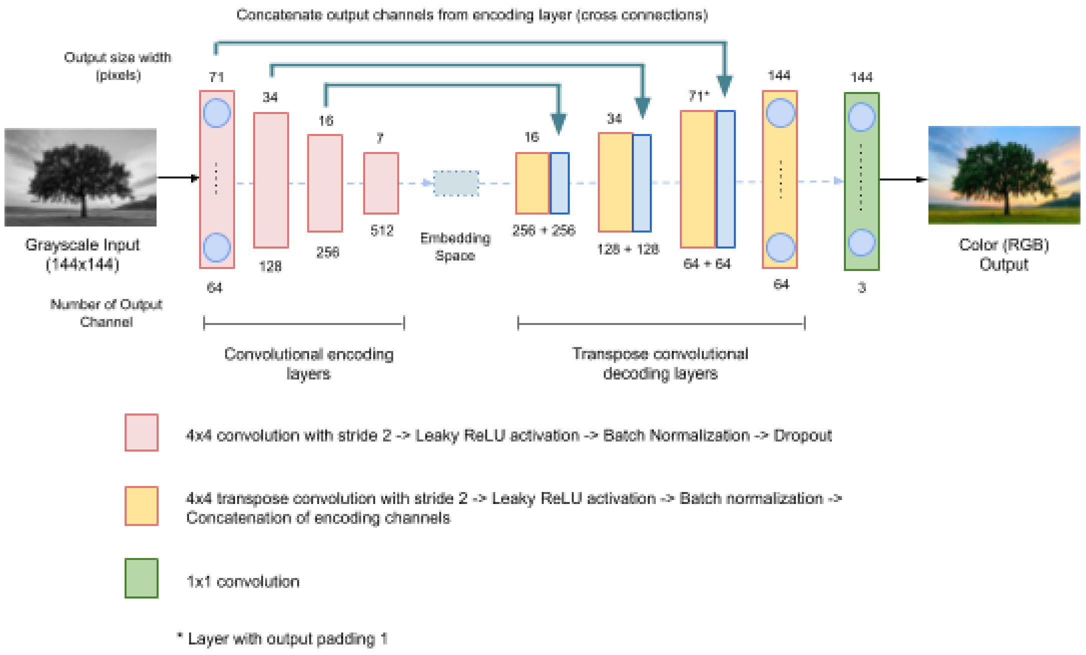

Concatenate output channels from encoding layer (cross connections)
Output size width (pixels)
71
34
16
7
16
34
71*
144
144
Grayscale Input (144x144)
64
128
256
512
Embedding Space
256 + 256
128 + 128
64 + 64
64
3
Color (RGB) Output
Number of Output Channel
Convolutional encoding layers
Transpose convolutional decoding layers
4x4 convolution with stride 2 -> Leaky ReLU activation -> Batch Normalization -> Dropout
4x4 transpose convolution with stride 2 -> Leaky ReLU activation -> Batch normalization -> Concatenation of encoding channels
1x1 convolution
* Layer with output padding 1

Figure 4*: The U-net Generator Network for the Conditional DCGAN.*

The Generator network was a U-net autoencoder with 4 convolutional layers and 4 transpose convolutional with a final convolutional layer to create the coloured image. The autoencoder learns the embeddings essential to colour the image. The cross-connections between the encoding and decoding layers ensure the preservation of locational and contextual features across the network bottleneck, allowing for better feature recreation while reducing the training time [12].

Discriminator Network

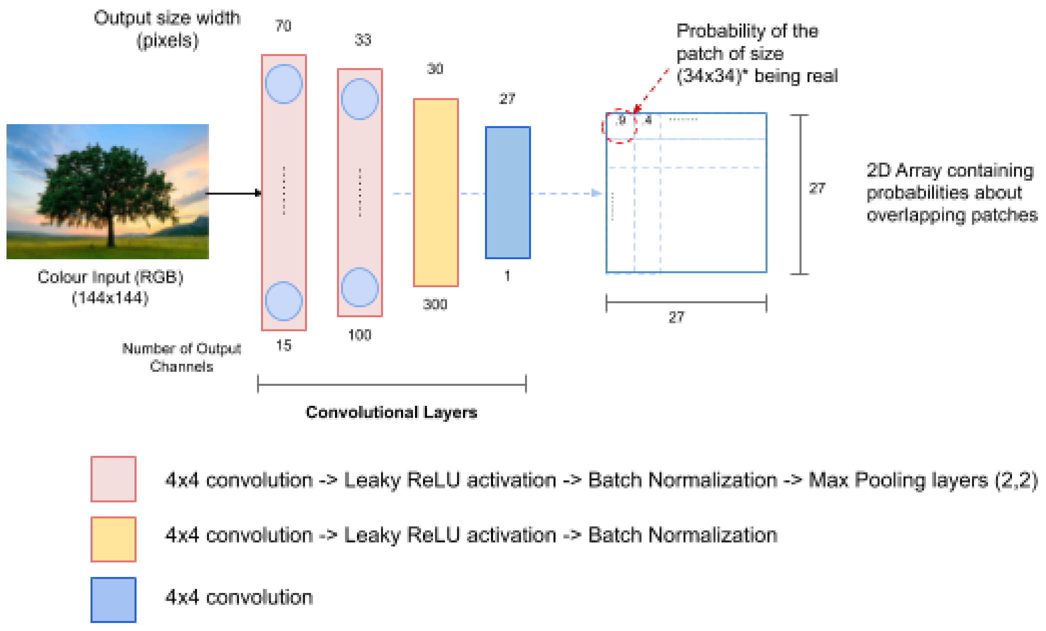

Output size width (pixels)
70
33
30
27
Probability of the patch of size (34x34)* being real
.9
.4
27
2D Array containing probabilities about overlapping patches
1
300
27
Colour Input (RGB) (144x144)
Number of Output Channels
15
100
Convolutional Layers
4x4 convolution -> Leaky ReLU activation -> Batch Normalization -> Max Pooling layers (2,2)
4x4 convolution -> Leaky ReLU activation -> Batch Normalization
4x4 convolution

***Figure 5**: The PatchGAN Discriminator Network for the Conditional DCGAN.*

The Discriminator network was a PatchGAN Convolutional Neural Network (CNN) with four convolutional layers. A CNN can extract essential features of the input image to classify it as fake or real. The output of the CNN was a grid of probabilities of 729 overlapping patches of the image being real. The patch size was 34x34 pixels and was calculated based on the Pix2Pix paper [11]. The goal was to fine-tune the generator's performance in each part of the generated image.

Training Losses

$$GLoss = BCE(Discriminator(GeneratedImages)) + \lambda \times MAE(GeneratedImages, RealImages)$$

$$DLoss = \frac{BCE(RealImages) + BCE(GeneratedImages)}{2}$$

***Figure 6**: Generator Loss "GLoss" and Discriminator Loss "DLoss".*

The Discriminator loss was calculated as the average of the Binary Cross Entropy (BCE) loss on the real and fake images. The Generator loss was determined by the Discriminator with an additional weight of the Mean Absolute Error (MAE) between the generated and the real images. The MAE factor helps the Generator tune its weights in the correct direction and train faster.

Batch normalization in both the networks normalized each layer's activations so that the next layer could learn in a faster and stable manner. The dropout layers helped the generator become more robust and prevent overfitting by dropping hidden units randomly, forcing other units to carry the weight by learning better. The use of Leaky ReLU activation solved the problem of vanishing gradients and helped in training the model faster.

Hyperparameters

Table 3: Chosen Hyperparameter Tuning

Parameter	**Value**
Learning Rate	0.00025
Batch Size	99
Lambda	80
Epochs	60

Baseline Model

The baseline model consisted of a simple autoencoder with five encoding and decoding layers respectively. There were three main hyperparameters: (1) batch size, (2) learning rate, (3) number of epochs. These parameters were set to 128, 0.0001 and 25 respectively. The model was trained within 1-10 minutes on the training data.

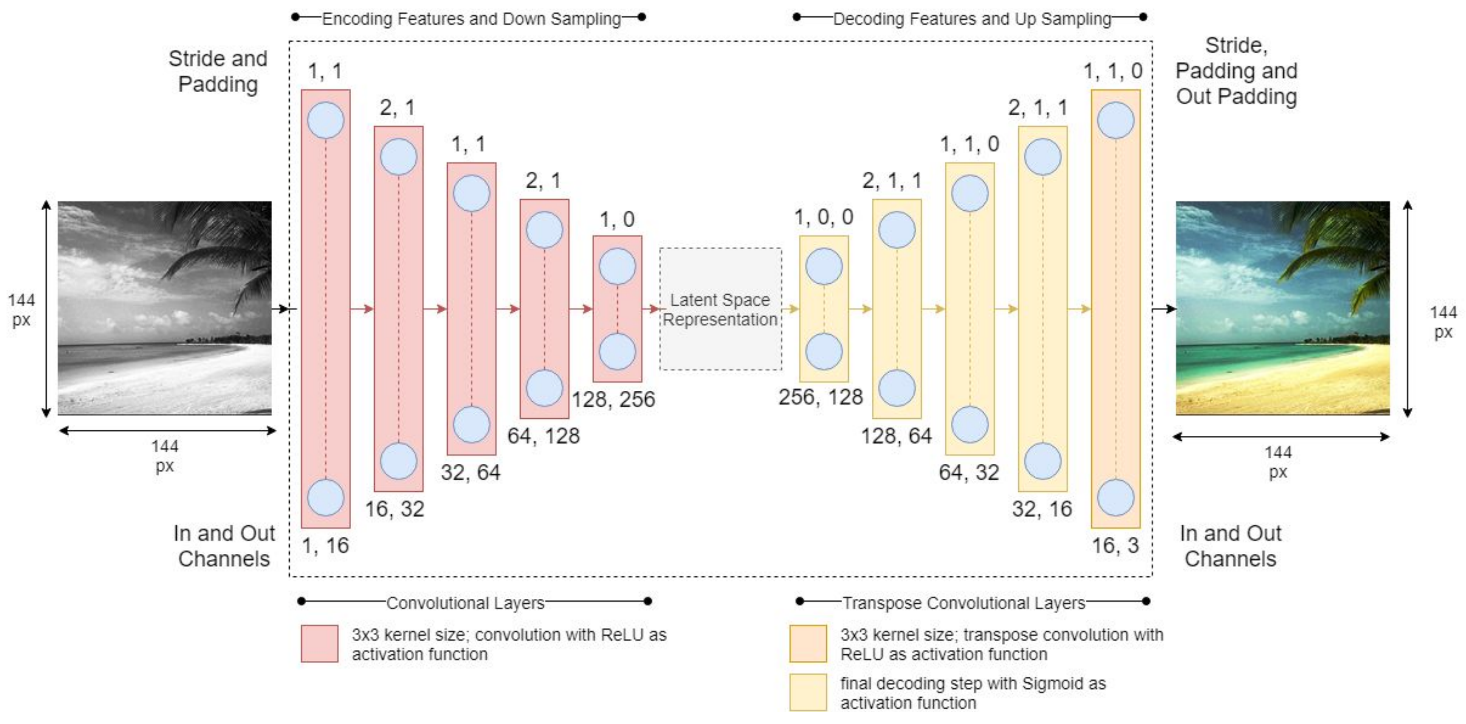

Encoding Features and Down Sampling
Decoding Features and Up Sampling
Stride and Padding
Stride, Padding and Out Padding
1, 1
2, 1
1, 1
2, 1
1, 0
1, 0, 0
2, 1, 1
1, 1, 0
2, 1, 1
1, 1, 0
Latent Space Representation
144 px
144 px
144 px
144 px
1, 16
16, 32
32, 64
64, 128
128, 256
256, 128
128, 64
64, 32
32, 16
16, 3
In and Out Channels
In and Out Channels
Convolutional Layers
3x3 kernel size; convolution with ReLU as activation function
Transpose Convolutional Layers
3x3 kernel size; transpose convolution with ReLU as activation function
final decoding step with Sigmoid as activation function

Figure 7: *Depiction of Baseline Model.*

Results

Quantitative

Primarily, since the project involves image recreation of 3 channel deep images (i.e., RGB), it is difficult to obtain the correct quantitative metrics to express the performance of the model. Our team opted primarily to rely on mean absolute error (MAE) as opposed to mean squared error (MSE) because MSE drastically punishes the network for more substantial errors. From our trials, MSE resulted in worse results because the model would try different weights to correct the large error values, often going towards a suboptimal minimum.

Ideally, the loss for both the generator and the discriminator in a generative adversarial network converge to similar values. The following graphs showcase the training loss on both the generator and the discriminator.

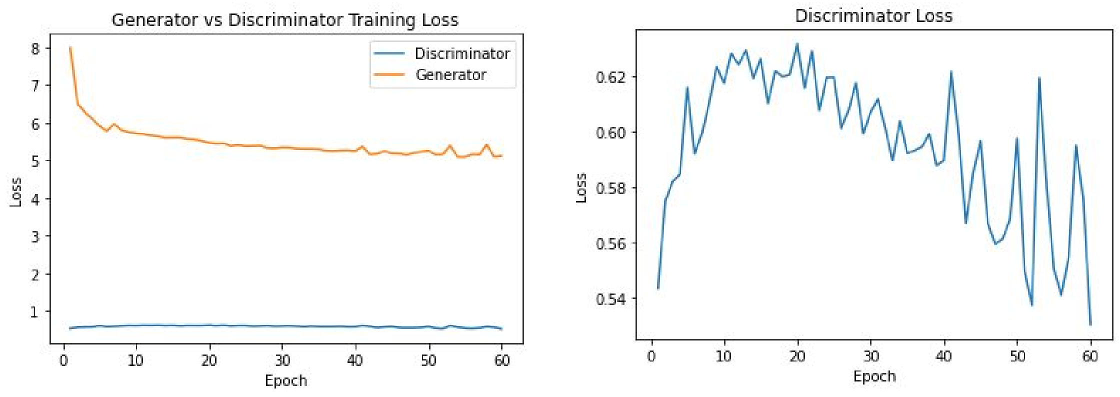


***Figure 8**: The generator and discriminator training loss. Discriminator loss zoomed in on the right graph..*

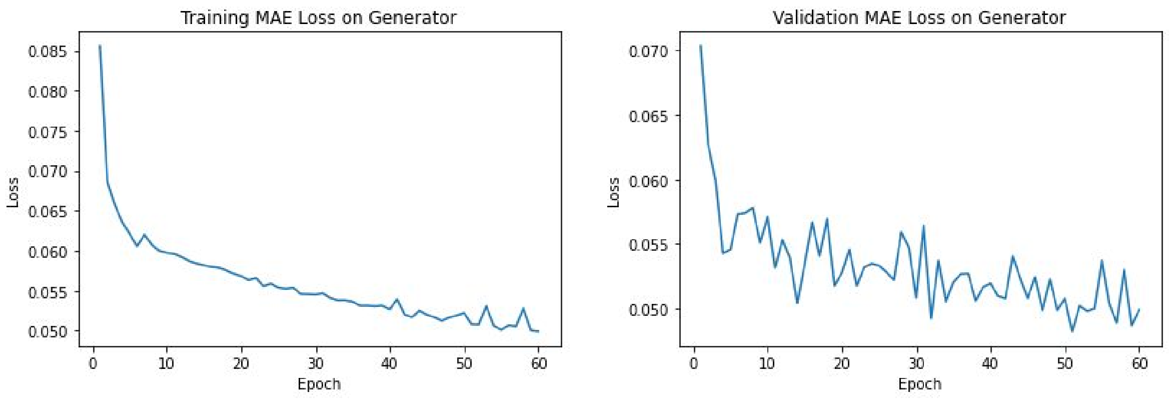


***Figure 9**: The Training and Validation MAE loss.*

Notice how the generator and discriminator loss do not converge to the same value, but the loss values are getting closer to each other. The reason might be that the proposed generator network might not be deep enough to account for this loss. However, even with this, the MAE loss is sufficiently small enough that the results are not very negatively affected. Furthermore, notice that the validation and training MAE losses both decrease, suggesting that the model is learning.

The model has the following results on testing and validation with the parameters mentioned above.

Table 4: Training and Validation MAE Loss

Testing Loss	Validation Loss
0.0519	0.0499

The above numbers seem quite low, suggesting that the output images should be almost identical. However, slight variations in colour pixel values result in a very different hue. As a result, our model can recolour the image similar to the original, but not perfect.

Qualitative

Since this is a structured learning problem involving images, qualitative results best showcase the models' ability to recolour pictures successfully. Visually, one can see that the reproduced images are very similar to the original image. There are slight deviations in colours, but that does not deteriorate the original mission of ColourizeIt!, which is to gain an insight into the past.

The following figures showcase a variety of training and validation images from the model. One or two images from each class are displayed in the figure.

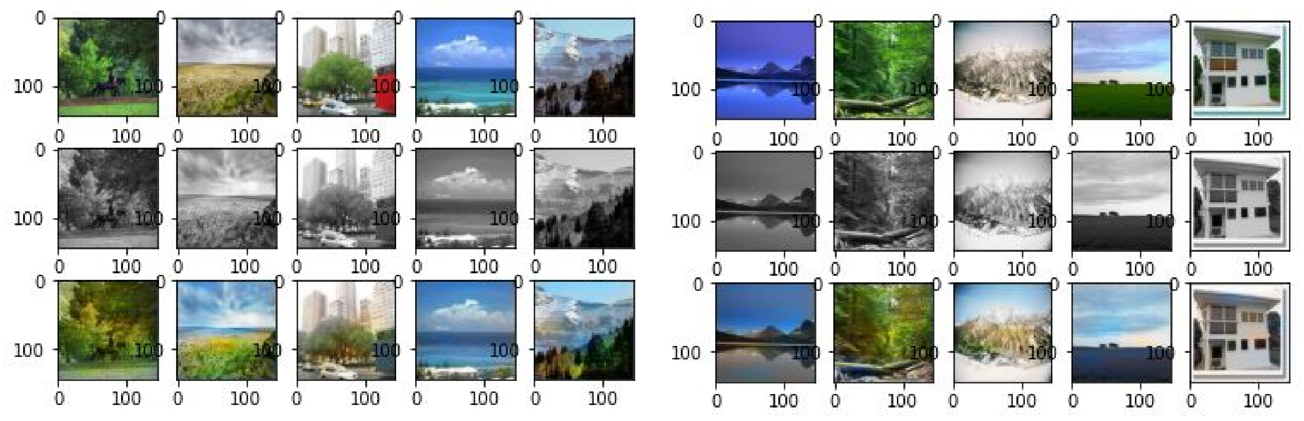


Figure 10: *Training images on the left side and validation images on the right. First row: Original image, last row: model output.*

The model colours the training greyscale images well because it was directly trained on those. The following figure demonstrates the capability of the model on various testing images from each of the classes. Further note, the team did not use these images in hyperparameter tuning.

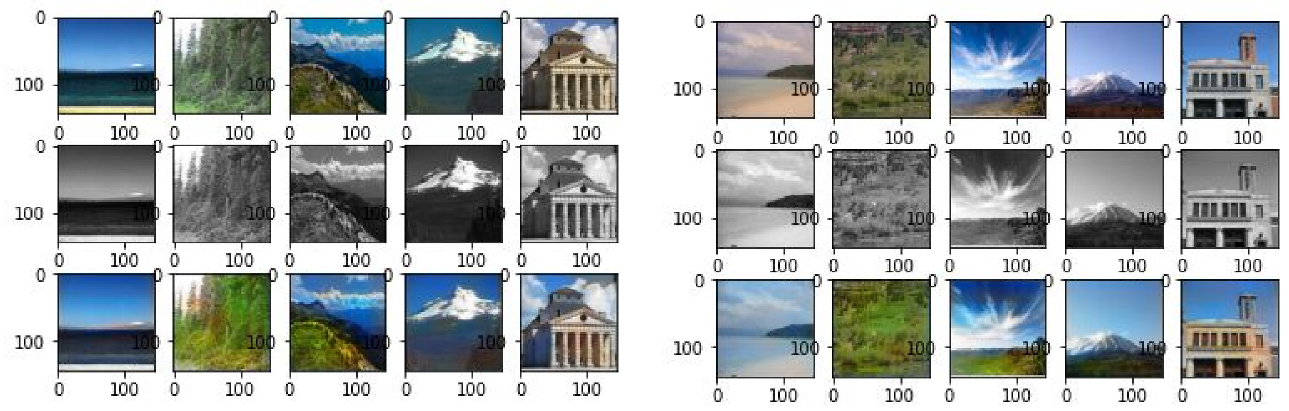


Figure 11*: Performance on Testing images from various classes. The top row is the original image. The last row is the model output.*

The model delivers a significant degree of accuracy on the testing images (refer to the appendix to see additional testing). But it is clear that there are certain patches in many images that are incorrectly coloured. For some images, this issue is escalated to a significant degree. For example, refer to the image below.

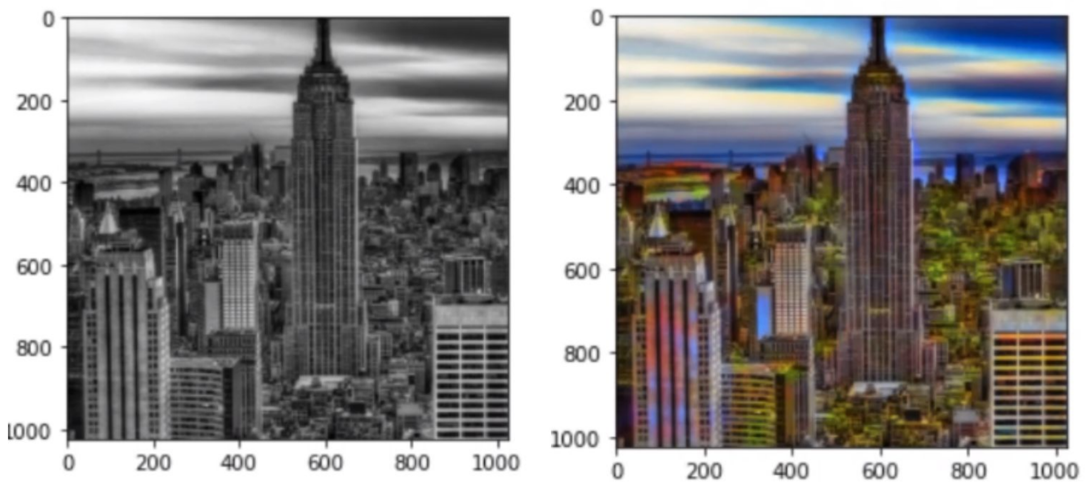


Figure 12: *Testing image showcasing the model’s weaknesses*

Here you can see that the model is incorrectly coloring many of the buildings yellow. The reason for this might be related to the chosen hyperparameters, the lambda value or the model depth. Because of the chosen lambda value, the model, when unsure, could be optimizing the weights based on a fixed color. We observed that the model coloured the patchy parts of the images differently depending on the lambda value. A solution for this might be to create a deeper generator network, which the current training hardware did not allow.

Discussion

Overall, the model performed well, given the hardware limitations that were encountered. Some images generated by the model, as shown in the Qualitative Results section, did have certain patches that were either incorrectly coloured or had a bias towards a certain hue. It was observed the model performed well on classes with well defined structures, such as buildings, as it was able to extract and learn the features effectively during the training process.

The primary model was much more complicated than the baseline model and performed much better. As shown below, the images produced by the primary model were brighter and had higher colour contrast. Moreover, the generated images from the primary model resembled the original images more closely than the images from the baseline model.

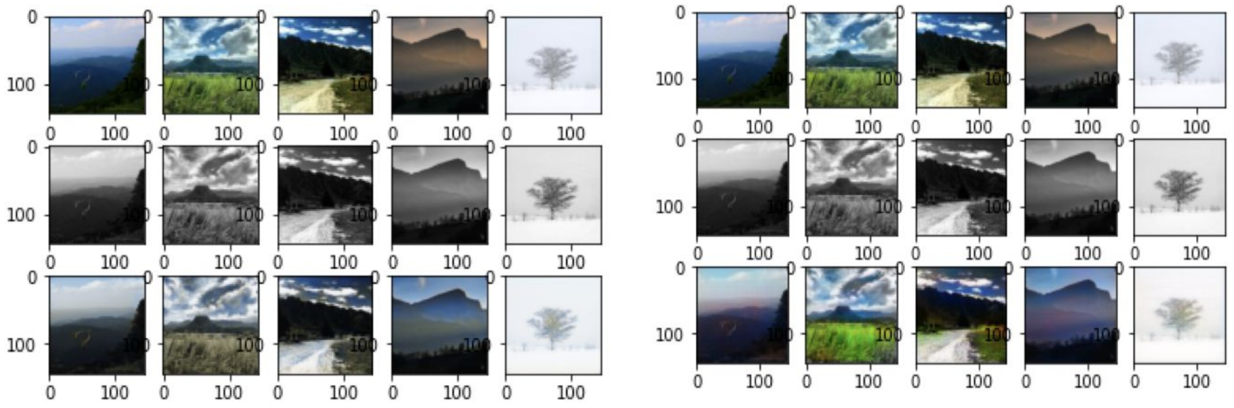
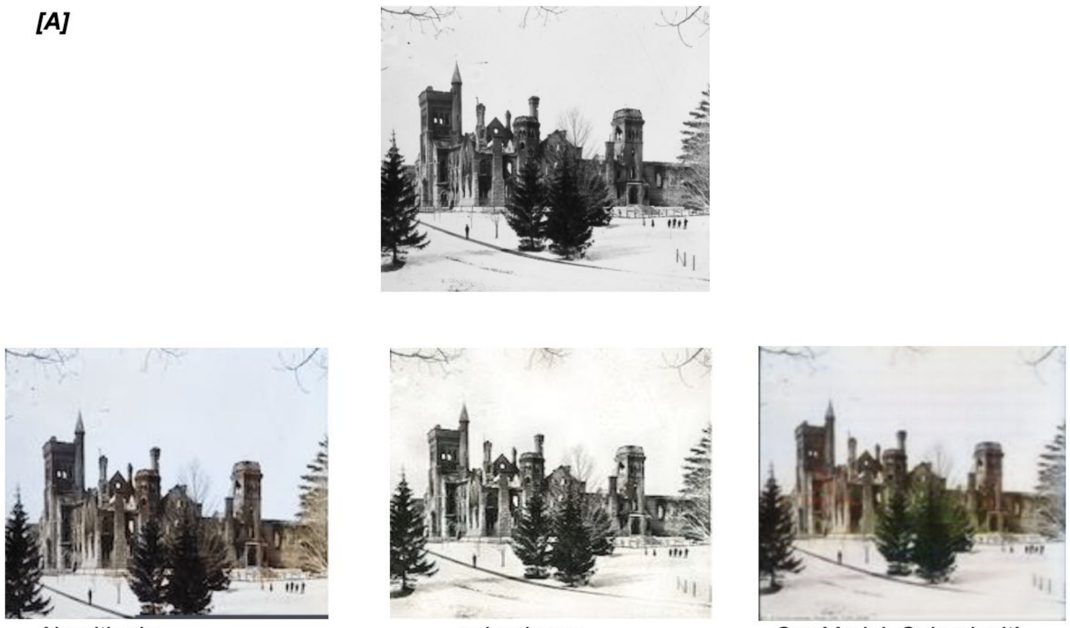


Figure 13: *Baseline results (left) and the GAN (right). The first row is the original image and the final row is model output.*

Comparing the primary model to Algorithmia and colourise.sg, it was seen that it performed relatively well, given the GPU limitations and a much smaller training dataset.

[A]



[B]

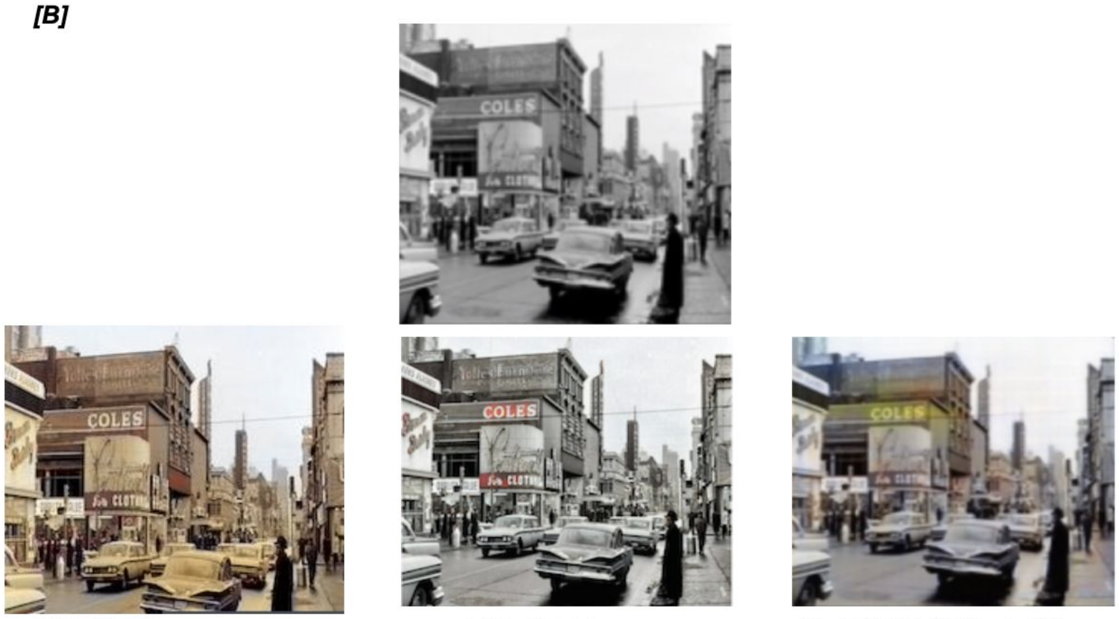


***Figure 14 A, B**: Comparing the same image(s) on all three models. The top (grayscale) image was tested on two state of the art models and ColourizeIt!*

The above examples illustrate how the models performed. Algorithmia is training on 1.3 million images from ImageNet and is expected to do well. However, its coloured output was heavily saturated as seen in Figure 14-B. Since colourise.sg trains on old images related to

Singapore, it produced an ancient filter effect in the colourized image in Figure 14-A. It failed to colourize the image in Figure 14-B as only a few patches of red are seen. ColourizeIt! not only recoloured the whole picture but also made it look natural, as seen by the results generated for both images from the past.

Although the model performed well, we plan to improve and generalize its performance across all kinds of images. We plan to implement the CIE LAB colour space [3]. Since it includes all the colours in the spectrum as well as that outside of human perception, thus making it the most exact way to represent colour [13]. With the use of a powerful hardware system, we also plan to construct a deeper neural network, as done in other research papers. To extend the applications of ColourizeIt!, we plan to train on a bigger dataset, and not just limit it to landscapes, to generalize the colouring process to all kinds of images.

Ethical Considerations

There are always ethical considerations to take into account when deploying AI projects. Specifically, the broad concept of image re-colourization can bring up some ethical issues within society. Re-colourizing photos from the 1950s era can be seen as disregarding the intent of the photographer and removing the long “history,” including cultural and geographic context, behind that picture. Also, the image re-colourizer introduces serious ethical issues when the image contains humans. Humans have different skin tones and therefore predicting skin tones for a human subject may be controversial. To avoid this particular ethical issue, we have limited our data to images without humans.

Project Difficulty / Quality

The intuitiveness of this project from an artificial intelligence standpoint seems simple, as one is training a neural network to learn from coloured images and reproduce similar results on a grayscale image. Through working on this project for the past two months, the team realized that it is not as easy as it sounds. Due to the GPU hardware limitations, the group decided to restrict the shape of the image tensors to [3, 144, 144] by resizing and cropping the original images as well as picking out the best pictures for optimal feature extraction and learning. The training, validation and testing split also added a layer of difficulty as the team tried to balance out the RGB colour spaces as it would eliminate bias towards one colour in our results.

Training our GAN model added many uncertainties as the loss values did not give much value in what direction to tune the hyperparameters. In general, the team went through a steep learning curve to train GANs by reading many research papers and finding the best model that would work with the problem we are solving. Instead of just reading them, the team decided to code the models and try to train accordingly to measure the best results. Finally, the U-net model was the one the team chose from the research papers. The implementation of this model was very time consuming and intricate to follow as there were many layer size calculations the team had to perform to even begin with the training, let alone the hyperparameter tuning.

After the numerous hurdles along the way, the results shown above came out exceptionally well and fared very well to the state of the art models such as Algorithmia and colourise.sg. The accurate outputs were a result of days of tuning hyperparameters and changing the generator model multiple times. All in all, the entire process was an enriching experience; seeing the model can produce grayscale images better than some models in research papers quickly became the team’s highlight of this project.

References

[1] S. Othman, “12 Fast-Growing Shade Trees,” *Arbor Day Blog*, 30-Jul-2018. [Online]. Available: https://arbordayblog.org/landscapedesign/12-fast-growing-shade-trees/. [Accessed: 22-Feb-2020].

[2] R. Zhang, P. Isola, and A. A. Efros, “Colorize Black and White Photos,” *Algorithmia*, 2017. [Online]. Available: https://demos.algorithmia.com/colorize-photos. [Accessed: 19-Feb-2020].

[3] R. Zhang, P. Isola, and A. A. Efros, “Colorful Image Colorization,” *Colorful Image Colorization*. [Online]. Available: http://richzhang.github.io/colorization/. [Accessed: 19-Feb-2020].

[4] A. Tan, P. Lim, and T. K. Wei, “ColouriseSG,” *ColouriseSG*. [Online]. Available: https://colourise.sg/. [Accessed: 19-Feb-2020].

[5] J. Antic, “DeOldify,” *GitHub*, 20-Feb-2020. [Online]. Available: https://github.com/jantic/DeOldify. [Accessed: 23-Feb-2020].

[6] P. Lim, “Data.gov.sg Blog,” *Data.gov.sg Blog*, 03-Feb-2019.

[7] “SuperParsing: Scalable Nonparametric Image Parsing with Superpixels,” *Welcome to the UNC Department of Computer Science - Computer Science*. [Online]. Available: http://www.cs.unc.edu/~jtighe/Papers/ECCV10/. [Accessed: 14-Mar-2020].

[8] A. ROUGETET, “Landscape Pictures,” *Kaggle*, 12-Aug-2019. [Online]. Available: https://www.kaggle.com/arnaud58/landscape-pictures. [Accessed: 15-Mar-2020].

[9] C. Valenzuela, “ml5js/ml5-data-and-models,” *GitHub*. [Online]. Available: https://github.com/ml5js/ml5-data-and-models/tree/master/datasets/images/landscapes. [Accessed: 19-Mar-2020].

[10] Z. Xu, “Architectural Style Classification using MLLR - Zhe Xu's Homepage,” *Google Sites*. [Online]. Available: https://sites.google.com/site/zhexuutssjtu/projects/arch. [Accessed: 05-Apr-2020].

[11] P. Isola, J.-Y. Zhu, T. Zhou, and A. Efros, “Image-to-Image Translation with Conditional Adversarial Networks,” *Image-to-Image Translation with Conditional Adversarial Networks*, 2017. [Online]. Available: https://phillipi.github.io/pix2pix/. [Accessed: 29-Mar-2020].

[12] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” *Cornell University* , May 2015.

[13] “Lab Color,” *MATLAB & Simulink*, 2020. [Online]. Available: https://www.mathworks.com/discovery/lab-color.html. [Accessed: 07-Apr-2020].

Appendix

More testing images showcased. Top row is original image and bottom row is model output.