IMAGE COLOURIZATION VIA CONVOLUTIONAL NEURAL NETWORKS AND DEEP LEARNING

Youssef Fikry
Student# 1005678901
youssef.fikry@mail.utoronto.ca

Harkirpa Kaur
Student# 1011242479
harkirpa.kaur@mail.utoronto.ca

Peter Leong
Student# 1005678901
peter.leong@mail.utoronto.ca

Thulasi Thavarajah
Student# 10115358424
t.thavarajah@mail.utoronto.ca

ABSTRACT

This template should be used for all your project related reports in APS360 course.
– Write an abstract for your project here. Please review the **First Course Tutorial** for a quick start —-Total Pages: 5

1 INTRODUCTION

While the colour photography processes first emerged in the 1890s, colour photography did not become widely accessible until the 1970s **?**. As a result, most historic photographs are black and white and lack the visual richness that modern viewers are accustomed to. In addition, individuals who have their cataracts removed as a part of vision restoration processes have shown to struggle with identifying objects in black and white images **?**, making most historic photographs inaccessible to them. This project aims to use deep learning to automatically colorize black and white images, with the goal of restoring visual information and making historical imagery more accessible for all audiences. Traditional, non deep-learning based colorization methods often produce desaturated results and rely heavily on human guidance **?**, making them non-scalable. Conversely, deep networks such as CNNs effectively capture spatial and semantic features and produce realistic colourized images without user interaction **?**, making deep learning an ideal approach for image colorization.

1.1 BACKGROUND & RELATED WORK

The challenge of image colourization has been approached in a wide variety of ways. Even within solutions involving deep learning techniques, there numerous unique design choices. One grouping method **?** results in five categories: simple colourization neural networks, user-guided colourization neural networks, diverse colourization neural networks, multi-path colourization neural networks, and examplar-based neural networks.

Simple colourization neural networks use feedforward CNNs to map grayscale inputs to colour outputs. One of the foremost solutions proposed by **?** used a fully convolutional network to predict the a and b channels of the CIELAB colour space from grayscale images. Their architecture is composed of several convulational layers, each followed by a ReLU activation function and a batch normailization layer.

User-guided colourization neural networks use user input to guide the colourization process. One such solution **?** uses a fully convolutional network to predict the a and b channels of the CIELAB colour space from grayscale images, but also takes user input in the form of user-provided colour scribbles. The network is trained to minimize the difference between the predicted and user-provided colours, allowing it to learn to colourize images in a way that is consistent with the user's input.

Diverse colourization networks produce multiple colourization outputs for a given grayscale input. One such solution **?** uses a generative adversarial network (GAN) to produce multiple colourization outputs for a given grayscale input. The GAN is trained to minimize the difference between the predicted and ground truth colours, allowing it to learn to produce diverse colourization outputs.

Multi-path colourization neural networks differentiate features at different scales. **?** proposed a multi-path colourization neural network that uses multiple convolutional layers to extract features at different scales. The network is trained to minimize the difference between the predicted and ground truth colours, allowing it to learn to produce colourization outputs that are consistent with the features at different scales.

Exemplar-based neural networks use a set of exemplar images to guide the colourization process. In **?**, example images are used to trasnfer the colour to the target image. Each instance is ouput to two different colourization networks which fuse to yield the final result. This group of solutions is easier to implement, as learning to colourize instances is significantly easier than learning to colourize an entire image.

2 METHODOLOGY

This section outlines our data processing pipeline, describes the architecture of the proposed model, and introduces the baseline model used for comparative evaluation.

2.1 DATA PROCESSING

The project's image dataset was compiled using individual datasets from Kaggle.com, an online platform that provides access to real-world datasets and a community for data scientists. To train a model that can generalize to a broader range of images, the final dataset for this project is comprised of three categories: human, animal, and scenic. All datasets chosen have a public domain license.

2.1.1 REPURPOSING ONLINE DATASETS

The image dataset was chosen because its original purpose is for human detection (**?**). As a result, the dataset contains a diverse range of 17,300 images of people in different environments.

The animal image dataset was originally for image classification as it contains 5,400 images of 90 different animals (**?**). For this project's purpose, this dataset is ideal as it results in a diverse set of images with an equal distribution of the number of images per animal.

The scenic image dataset from **?** contains 4,319 images of a variety of landscapes that encompass a large breadth of colour palettes, which can impact how well the final model performs.

2.1.2 CLEANING UP THE DATASETS

All the images from each dataset were extracted from their original folder organizations and re-located into a folder corresponding to their category. Due to the disparity in the size of the three datasets, each dataset was reduced to exactly 4200 images using Python's `random.sample` function with the seed set to 42. The images were then renamed in accordance to their respective categories (ex. human_0001.jpg).

2.1.3 FORMATTING THE DATA

This project requires a unique dataset with black and white images, along with their coloured counterparts. To format the cleaned up data and create this dataset, PyTorch's `torchvision.transforms` library was utiltized. The three sets of images were converted to 256 x 256 pixel ground truth images using the `Resize` transform and then transferred into a folder corresponding to their category. Following this, `Grayscale` transform with a output channel of 1 was used to convert the 256 x 256 pixel colour images to 256 x 256 pixel grayscale images to feed into the model.

2.1.4 Splitting Dataset Into Training, Validation and Test Sets

To create the training, validation and test sets, a ratio of 70:15:15 was chosen. The first 2940 images of each of dataset categories (human/animal/scene) were relocated to the training set. The remaining images were split evenly to create the validation and testing datasets.

2.1.5 The Final Dataset

The final dataset is composed of 12,600 pairs of color and their corresponding grayscale images. Each category (human, animal, and scenic) contains 4,200 image pairs. There are a total 8,820 pairs in the training set, and the validation and test sets each contain 1,890 image pairs. An even distribution of each category was maintained in the training, validation and test sets.

2.2 Illustration

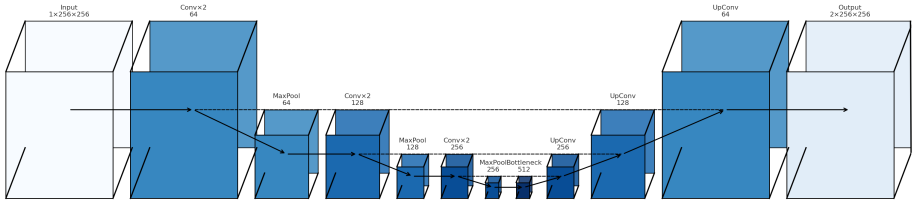

Figure 1: Encoder–decoder colorization network with skip connections.

The top row is the encoder (Conv ×2 → MaxPool blocks) that downsamples the grayscale L channel from 256×256 to 32×32 while increasing feature depth (64 → 512). The bottom row is the decoder (UpConv blocks) that upsamples back to full resolution and predicts the two chroma channels (a, b). Dashed arrows show the three skip connections that pass fine-grained features from encoder to decoder stages.

2.3 Architecture

We propose a convolutional encoder–decoder network for the colorization task, which is a common and effective choice for image-to-image translation problems. **?** The model will operate in the CIE Lab color space: the input is a grayscale image (the L channel), and the network is trained to predict the two chrominance channels (a and b) as output. After prediction, the input L channel and the output a,b channels are combined and converted back to an RGB image for visualization. Using the Lab space decouples intensity from color, which aligns with human vision and generally produces more realistic results than direct RGB prediction. **?**

The architecture itself is an encoder–decoder CNN resembling a simplified U-Net. The encoder consists of several convolutional layers (with ReLU activations and batch normalization) that gradually downsample the image, extracting higher-level features as the spatial size reduces. For example, we might start with a 3x3 conv layer with 64 filters, followed by another conv layer with 128 filters, each followed by a pooling (or stride-2 conv) to halve the spatial dimensions. After a few such layers, a bottleneck layer (e.g. a 3x3 conv with 256 filters) will capture abstract features of the grayscale input. The decoder then mirrors this process: it uses upsampling layers (e.g. Conv2DTranspose or nearest-neighbor upsampling) and additional conv layers to upsample the feature maps back to the original image resolution while predicting the a,b color channels. We plan to include skip connections between matching encoder and decoder levels (as in U-Net) so that fine-grained details from early layers (edges, textures) are directly passed to the corresponding decoder layers. These skip connections should help the network recover details and prevent blurry outputs by combining low-level and high-level features. **?**

To balance performance and simplicity, we will leverage transfer learning for the encoder. In particular, we are considering initializing the encoder with a pretrained backbone like the first few layers of ResNet-18 or VGG16 (trained on ImageNet). This gives the model a head-start in recognizing

semantic features—useful given our dataset of human and animal images, where recognizing objects like faces, fur, or backgrounds can inform color choices. **?** Using a pre-trained encoder should enable the network to learn meaningful colorization with fewer training examples and epochs. **?** The decoder and any additional layers will be initialized randomly and learned from scratch. We will train the network end-to-end to minimize a pixel-wise loss between the predicted and ground-truth color channels (e.g. mean squared error in Lab space).

It's worth noting that some prior works frame colorization as a classification problem over discrete color bins to better capture the ambiguous nature of predicting color. For instance, Zhang et al. (2016) predict a probability distribution over possible a,b values for each pixel instead of regressing exact values. **?** While such techniques (often coupled with class re-balancing or perceptual losses) can produce vibrant results, they add complexity beyond the scope of a course project. In our architecture, we favor a straightforward regression approach – the network directly outputs continuous a and b values – which is simpler to implement and sufficient for plausible colorization.

2.4 Baseline Model

As a baseline, we will start with a very simple colorization approach against which to compare our full model. One trivial baseline is to output the input grayscale image as-is in color (i.e. replicate the L channel into RGB), which yields a degenerate colorization. This sets a minimum benchmark – any learning model should outperform simply producing a monochrome image.

For a more meaningful baseline, we will implement a shallow convolutional network inspired by basic encoder–decoder examples. **?** This baseline model will take the grayscale L channel as input and produce a,b chrominance channels, but it will have only a small fraction of the capacity of our main architecture. For instance, the baseline could use just two convolutional layers for encoding (e.g. 64 filters and 128 filters) followed by a single upsampling decoder to reconstruct the output colors. Concretely, the image might pass through Conv2D(64) and Conv2D(128) layers (with ReLU), then a pooling layer to downsample; at that point a minimal "bottleneck" conv layer can be applied, and finally an upsampling (Conv2DTranspose) step generates the 2-channel output map. **?** This simplistic model does not incorporate any skip connections or pretrained knowledge, and it has far fewer parameters than the proposed full model. We expect its predictions will capture only basic correlations (for example, mapping lightness to a bland average color) and often look desaturated or unrealistic. **?**

Such a baseline provides a useful point of comparison: it represents what a rudimentary learning method can achieve without much capacity or context. By evaluating our advanced model against this baseline, we can quantify the gains from using a deeper architecture and more sophisticated design. If the baseline's output is dull or mostly grayish (as often seen in naive colorization attempts), whereas our full model produces more vibrant and context-appropriate colors, it will demonstrate the effectiveness of the chosen architecture. **?**

3 Ethical Considerations

The dataset being used is public, so there are no copyright or consent issues. However, the dataset may contain racial or demographic imbalances, which could cause the model to generalize poorly or be biased towards specific skin tones. This may result in racially inaccurate or culturally insensitive outputs. A similar behaviour may be observed with animals, where a lack of diversity in breeds or fur colours in the dataset can result in misleading results. If the outputs produced by the model are used in educational contexts or in breed identification, they can contribute to misinformation. Furthermore, since the model results are plausible but cannot be verified, there is a risk that users may overtrust the outputs in sensitive contexts.

4 Project Plan

Our high-level project planning has been completed using a Gantt chart seen below. It contains many of the detailed tasks we have completed and will complete in the future. This easily shows who is responsible for each task, the time it will take to complete, and the dependencies between tasks. Generally, our internal deadlines have been set at least a few days prior to the official deadlines to

ensure we have time to review our work and make any necessary changes. In the event that a new member joins the team, they can easily refer to this chart to see what responsibilites they can take on and what tasks are still available. Our Gantt chart can be found in Appendix A.

4.1 Team Communication & Coordination

We intend to use Discord as our primary communication platorm, Google Drive for file sharing (Colab notebooks, datasets, etc.), and GitHub for version control. To ensure we do not overwrite each other's work, we will use branches for each feature or task, and merge them into the main branch once they are complete. We will also have weekly meetings to discuss our progress, any issues we are facing, and plan for any action items for the subsequent meeting. Currently, we intend to meet on Saturday mornings at 8:00 AM EST as some of our team members are in different time zones.

5 Risk Register

6 Appendix A: Gantt Chart

PROJECT TITLE
APS360: Bridge Classification
DATE
06/13/2025
TASK NUMBER
TASK TITLE
START DATE
DUE DATE
DURATION (DAYS)
TEAMMATE RESPONSIBLE
PHASE ONE
PHASE TWO
PHASE THREE
PHASE FOUR
WEEK OF JUN 2
WEEK OF JUN 9
WEEK OF JUN 16
WEEK OF JUN 23
WEEK OF JUL 7
WEEK OF JUL 14
WEEK OF JUL 21
WEEK OF JUL 28
WEEK OF AUG 4
M T W R F