Finding the Best Architecture for Fully-Automated Image Colorization

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Gabriel Nkole
2377848

Supervised by Dr Richard Klein

May 16, 2023

A proposal submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Bachelor of Science with Honours

Abstract

A variety of architectures can be used for deep learning colorization models and currently there is no concrete review that determines which architecture is the best and why. Therefore, we will undergo an analysis of the different architectures and find the one best suited for colorization after considering a range of metrics. We hypothesize that the diffusion model is the best architecture under all circumstances except speed.

Declaration

I, Gabriel Nkole, hereby declare the contents of this research proposal to be my own work. This proposal is submitted for the degree of Bachelor of Science with Honours in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Acknowledgements

Thank you to my supervisor Dr. Richard Klein who provided guidance on how to approach this project.

Contents

Preface
Abstract . . . i
Declaration . . . ii
Acknowledgements . . . iii
Table of Contents . . . iv
List of Figures . . . vi

1 Introduction **1**

2 Background and Related Works **2**
2.1 Introduction . . . 2
2.2 Background . . . 2
2.2.1 Convolutional Neural Network (CNN) . . . 2
2.2.2 Generative Adversarial Network (GAN) . . . 3
2.2.3 Diffusion Model . . . 3
2.2.4 Transformer . . . 4
2.3 Related Works . . . 4
2.3.1 PixColor: Pixel Recursive Colorization (CNN) . . . 4
2.3.2 Unsupervised Diverse Colorization via Generative Adversarial Networks (GAN) . . . 5
2.3.3 Instance-aware Image Colorization (CNN) . . . 5
2.3.4 Colorization Transformer (Transformer) . . . 6
2.3.5 Palette: Image-to-Image Diffusion Models (Diffusion Model) . . . 6
2.3.6 PalGAN: Image Colorization with Palette Generative Adversarial Networks (GAN) . . . 7
2.4 Performance Evaluation . . . 7
2.4.1 Quantitative . . . 7
2.4.2 Qualitative . . . 8
2.5 Conclusion . . . 8

3 Research Methodology **9**
3.1 Problem Statement . . . 9
3.2 Hypothesis . . . 9
3.3 Research Questions . . . 10
3.4 Methodology . . . 10

3.4.1 Research Design 10
3.4.2 Methods 11
3.4.3 Dataset 14
3.4.4 Implementation Details 14
3.4.5 Performance Evaluation 14
3.4.6 Limitations 15

4 Work Schedule **16**

5 Conclusion **17**

References **20**

List of Figures

2.1 Leaderboard for FID-5K colorization [Code 2022] 7

3.1 Maximum Likelihood Formula [Taboga 2021] 11
3.2 Log Likelihood Formula [Taboga 2021] 11
3.3 Palette Algorithms [Saharia *et al.* 2022] 12
3.4 Coltran Structure [Kumar *et al.* 2021] 13
3.5 Conditional Transformer Components [Kumar *et al.* 2021] 13
3.6 ColTran Objective Function [Kumar *et al.* 2021] 13
3.7 cGAN Objective Function [Isola *et al.* 2017] 14
3.8 L1 Loss [Isola *et al.* 2017] 14
3.9 PixColor Objective Function [Isola *et al.* 2017] 14

Chapter 1

Introduction

Photography was invented in the early 19th century, but the technology was very limited and could only produce black-and-white (monochrome) photographs. If it was necessary for a photograph to be in color, somebody needed to color it manually; this required expertise and was a time-consuming process [Li *et al.* 2020].

Although color photography became available to the public in the 1930s (theoretically eliminating the need for monochrome photography), it was not widely adopted until the 1960s and thus there was still a need for images to be colorized [Peerspace 2015].

With the advent of computer technology in the mid-20th century, automated processes for colorization were introduced. However, colorization has been difficult to perform with computers because it is an ill-posed problem. The only available information about a pixel is its intensity and this value needs to be used to predict all 3 RGB values (or the 2 other values in CIELAB space).

Many models have been proposed with varying success, and for this project the deep learning models will be analyzed. Within the relatively few years models of this type have existed (the first was introduced by Cheng *et al.* [2015]), they have shown great promise in solving this problem. Since there are several suitable neural network architectures for such models, it is necessary to compare them to find the one best-suited for colorization. Diffusion models have been performing well for many image tasks (including colorization) and we aim to prove that they are the overall best architecture for colorization.

With color photography being mainstream, the general public's need for colorization has mostly vanished. Today, it is commonly used to restore historic black-and-white photographs and videos, enhance medical images that inherently lack color or (simply) make grayscale images more appealing.

Chapter 2

Background and Related Works

2.1 Introduction

There are a variety of architectures to choose from when designing a deep learning colorization model. The 4 ubiquitous ones are: Convolutional Neural Networks, Generative Adversarial Networks, Transformers & Diffusion Models. This review explains these architectures in the Background section, discusses some important colorizations models in the Related Works section by chronological order and then evaluates said models in the Performance Evaluation section.

Although CNNs can exist as components within the other models, “CNN” on its own will refer to pure CNN models.

2.2 Background

2.2.1 Convolutional Neural Network (CNN)

A CNN is designed to simplify processing by learning weights on high-level features instead of directly on each input element. They mainly consist of three types of layers: the convolution layer, the pooling layer, and the fully connected layer. The convolution layer applies a filter on the input to produce a feature map. The pooling layer applies a filter on the input to reduce its dimensionality (and thus its complexity). The fully connected layer learns the relationships between features in order to achieve the desired output. These layers can be stacked to learn more complex features [IBM nd].

CNNs work well on images because they can more easily learn high-level features that assist in analyzing the image. Moreover, learning weights on the important features is less computationally expensive than on the individual pixels of an image.

2.2.2 Generative Adversarial Network (GAN)

GAN was first introduced in the research paper “Generative Adversarial Nets” by Goodfellow *et al.* [2020]. A GAN contains two sub-neural networks, a generator and a discriminator. The generator creates fake data, and the discriminator differentiates the fake data from the real data. The generator’s purpose is to deceive the discriminator, and these networks are both trained competitively.

The competitive nature of training between the generator and discriminator has resulted in GANs that can create impressive results. Consequently, GANs became the state-of-the-art model for many fields at the time, one of the most prominent ones being Vision. A common use case is the synthesizing of new images from noise (such as human faces, animals, etc.), and it has become an important model for image colorization by deep learning. GAN-based colorization models can produce realistic results able to fool human observers.

Although GANs have shown promise, the main challenge they face is their instability during training. Since the model is inherently adversarial, issues can arise where the model fails to converge or the generator cycles through a small set of outputs (mode collapse); however, there are techniques available to remedy these issues [Brownlee 2019].

2.2.3 Diffusion Model

The Diffusion Model was first introduced in “Deep Unsupervised Learning using Nonequilibrium Thermodynamics” [Sohl-Dickstein *et al.* 2015]. Historically, a difficulty for probabilistic models was how to design a model that can be evaluated and easily fit to the data while still being able to fit structure in arbitrary data. Achieving both was difficult as there was a trade-off between the two designs.

This paper introduces a method that uses a Markov Chain to gradually convert a simple distribution (e.g. Gaussian) to a target distribution by way of diffusion. The probabilistic model is explicitly defined at the end of the chain and because each step has an evaluable probability, this translates to the entire chain having this property [Sohl-Dickstein *et al.* 2015].

The model is defined by two processes: an iterative forward diffusion process and a reverse diffusion process. In the forward process, a small amount of noise is iteratively added to an element of the target distribution until T diffusion steps where the example fully becomes an element of the noise distribution. The reverse process then attempts to iteratively transform this noise back into the desired output by way of a denoising model. For this to be possible it is necessary to be able to find the element’s state at the previous timestep by only using its state at the current timestep, however, this transition function is initially unknown as it requires the entire dataset. Therefore, a model of the probabilities that can facilitate this is learned [Weng 2021].

Although the process of turning noise into the desired output is longer than in other generative models, learning the target distribution in small steps instead of directly (as other generative models do) allows the model to better learn the target distribution.

This can contribute to better results in any of the tasks that diffusion models are used for (such as image colorization).

Although this type of model has existed for several years, its potential to produce high quality results in Vision tasks has only been realized in recent years.

2.2.4 Transformer

Natural Language Processing (NLP) is a branch of Computer Science focused on analyzing and solving problems related to natural language data (text translation, text classification, etc.). The previous state-of-the-art architecture in the field was Long short-term memory (LSTM); however, it suffered from bad performance and failed to capture long-term dependencies between data [Vaswani *et al.* 2017].

In the 2017 research paper "Attention is All You Need" [Vaswani *et al.* 2017], a novel Neural Network architecture named the Transformer was proposed. As the name suggests, the model is based solely on the mechanism of attention, and the neural networks employed are simple feedforward networks. Attention is not a new concept, LSTMs already made use of attention by storing the previous state(s) of the network and having it influence the new state, however, Transformers rely on self-attention to determine long-term dependencies between tokens better than LSTMs. Moreover, the focus on self-attention has allowed for more parallelism and thus better performance [Vaswani *et al.* 2017].

It was evident that Transformers outperformed previous models in NLP tasks with far less training time, making it the new state-of-the-art in that field. Although purposefully built for NLP tasks, Transformers have found use in the field of Vision on problems such as image classification, object detection, super-resolution and (more importantly) image colorization [Khan *et al.* 2022].

2.3 Related Works

2.3.1 PixColor: Pixel Recursive Colorization (CNN)

Aside from Colorful Image Colorization (CIC) [Zhang *et al.* 2016], many CNN-based colorization models at the time produced desaturated, washed-out colorizations. This is because they were encouraged to predict the average color of the Ground Truth (GT) image. The loss in PixColor is instead based on log-likelihood estimation (unsupervised learning) to fix this problem and produce consistent results [Guadarrama *et al.* 2017]. PixColor and other models such as conditional Invertible Neural Network (cINN) [Ardizzone *et al.* 2019] have established log-likelihood estimation as a method for producing state of the art colorizations with CNN-based models.

PixColor consists of two CNNs. The first CNN, PixelCNN, is conditioned on a grayscale image to produce a low-resolution color image. The second CNN is conditioned on the output of PixelCNN and the grayscale image to produce the final high-resolution color image.

2.3.2 Unsupervised Diverse Colorization via Generative Adversarial Networks (GAN)

This paper by Cao *et al.* [2017] is one of the first (and most prominent) to propose a cGAN-based colorization model. A traditional GAN is insufficient for colorization because it is designed to only receive noise as input (and it is necessary to input a grayscale image so that a color image can be produced). Seeing that there may be a need to input some pre-existing data into the network, Goodfellow *et al.* [2020] also proposed the conditional GAN (cGAN) architecture, which accepts pre-existing data in addition to noise to generate output. This variant is what is used for all GAN colorization models.

While other image generators consist of convolutional and de-convolutional components, this model only contains convolutional layers with stride set to 1 to prevent the data shape from downsizing. This allows the layer shape to remain consistent, providing the possibility to feed the conditional information to the generator at every layer (multi-layer conditional). Training is still possible because the model learns its own up-sampling/down-sampling method, and the network architecture becomes simpler. This change in the generator's architecture is the main difference between this model and the others at the time. Otherwise, other components of the architecture remain mostly the same [Cao *et al.* 2017].

Comparisons were made on RGB vs YUV, single-layer noise vs multi-layer noise and single-layer conditional vs multi-layer conditional. The results showed that the best network configuration was YUV combined with multi-layer noise and conditional. The multi-layer noise especially contributed to more diverse colorizations and the multi-layer conditional contributed to stability [Cao *et al.* 2017].

The main reason Cao et al. chose GAN over CNN was that the CNN-based models at the time often produced brownish, desaturated images. These models were trained by comparing the output colors to the GT colors and designing a loss function around the differences (supervised learning) [Cao *et al.* 2017].

2.3.3 Instance-aware Image Colorization (CNN)

Deep learning methods often learned pixel/feature/image-level semantic information to produce colorizations. While such models can produce plausible, vivid colorizations, they are prone to produce results that contain color bleeding. This is because these models mostly capture pixel or image-level semantics and not object semantic information. Although these models work well when there are few objects in an image, they often worsen when there are several objects in a scene due to the above factors [Su *et al.* 2020].

InstCol aims to solve this problem by leveraging an off-the-shelf object detector to accurately capture object semantic information and create pleasing colorization results.

2.3.4 Colorization Transformer (Transformer)

Although the transformer architecture has existed since 2017 and has been used for many Vision tasks since then, the first ever image colorization transformer model “Colorization Transformer” (ColTran) was proposed in 2021.

A core idea of this model is the use of a concept called axial attention, which was adopted from the Axial Transformer [Ho et al. 2019]. Instead of directly computing global self-attention between all tokens of the input like the standard Transformer, local self-attention is computed separately on each axis of the input and hierarchically combined to compute the desired global self-attention. Only working on a subset of the input at a time in this manner decreases computational cost while preserving the relationships between tokens. Another concept leveraged is conditioning. Aside from the grayscale image, a given pixel of the image is also conditioned on all previous pixels (as per raster order) and this serves to increase the performance of the model. To make conditioning possible, the core Transformer components (Self-attention, MLP, Layer Normalization) need to be converted to their conditional counterparts [Kumar et al. 2021]. ColTran consists of 3 core components: ColTran Core, the Color Upsampler and the Spatial Upsampler. The general idea behind the model is to first produce a coarse coloring for a downscaled version of the grayscale image and then upsample the color and spatial of that image to produce a highly detailed colorization of the original image with the correct dimensions [Kumar *et al.* 2021].

The core itself consists of the grayscale encoder, outer decoder, and inner decoder subcomponents. The grayscale encoder learns new representations for all the pixels in the input. It is ran once per channel and information for each channel is encoded independently. The outer decoder then computes a state for all previous rows of a given pixel and the inner decoder computes a state for all previous pixels in the current row. The output of the core is a downscaled colorized 3-bit RGB image. The color upsampler then receives said image and the grayscale image and outputs a new 8-bit representation of the downscaled image. Finally, this result is first stretched to the original shape and is input alongside the grayscale image to the spatial upsampler to produce the high-quality colorized image [Kumar *et al.* 2021].

2.3.5 Palette: Image-to-Image Diffusion Models (Diffusion Model)

Palette is a unified framework for image-to-image translation, and in the original paper, was tested on four challenging imaging tasks namely: inpainting, uncropping, JPEG restoration and (most importantly) colorization [Saharia *et al.* 2022]. The framework is based on conditional diffusion models and no task-specific tuning or completely novel techniques were employed.

When trained for image colorization, Palette achieved a lower FID score than any other model. Thus, making it the current state-of-the-art model.

2.3.6 PalGAN: Image Colorization with Palette Generative Adversarial Networks (GAN)

PalGAN is a new colorization approach that uses palette estimation and chromatic attention to address the issues of color bleeding and multi-modal ambiguity in colorization. The rationale of PalGAN is to model a palette histogram from the image and assign correct colors from the palette to each pixel. The GAN consists of a palette generator, a palette assignment generator, and a color discriminator [Wang *et al.* 2022].

According to the paper, it has achieved better performance than ColTran on both qualitative and quantitative metrics. At the time of writing, this paper is still exceptionally new and has not yet been thoroughly discussed and reviewed.

2.4 Performance Evaluation

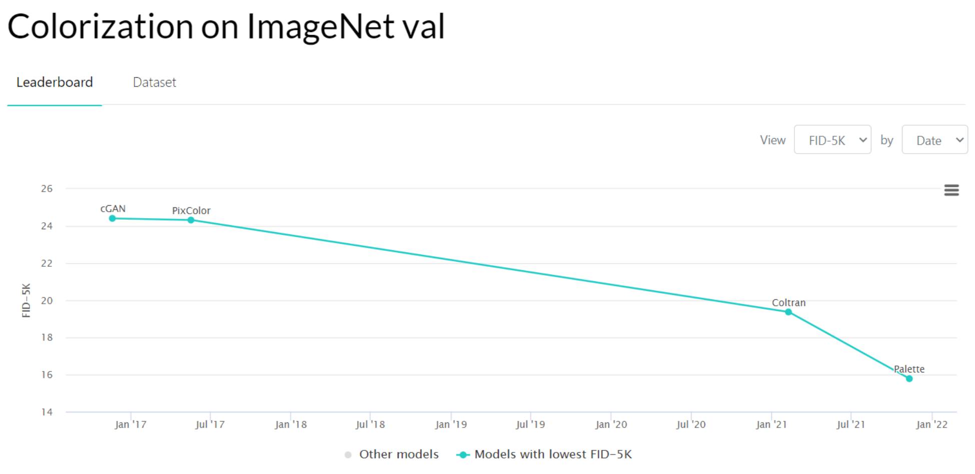


Figure 2.1: Leaderboard for FID-5K colorization [Code 2022]

2.4.1 Quantitative

The main quantitative measure of colorization quality employed was the Fréchet Inception Distance (FID). The lower the score, the higher the similarity to the GT image. In the Colorization Transformer paper, the FID scores for cGAN [Cao et al 2017], PixColor, ColTran and Palette on 5000 256x256 ImageNet images were computed to be 24.41 ± 0.27, 24.32 ± 0.21, 19.37 ± 0.09 and 15.78 respectively with the GT at 14.68 ± 0.15. PixColor and cGAN are almost indistinguishable from each other, ColTran is significantly better than both and Palette is significantly better than all previous models in terms of performance.

Therefore, in terms of quantitative performance, the order of the models is: Palette, ColTran, PixColor and cGAN. It is unknown where InstCol would rank.

2.4.2 Qualitative

The main qualitative measure of colorization quality employed was the Turing Test. Most notably, the fool rates for PixColor, Coltran & Palette were 29.9%, 36.55% & 47.8% respectively, meaning that Palette is close to the ideal fool rate of 50%.

2.5 Conclusion

This review sought to find which of the four architectures was best suited for colorization, and Palette produced the highest results in both quantitative and qualitative measures. The amount of training steps and the inference time was specified for some of the models, but the training time was not.

A problem when looking at the papers for these models is they do not all share the same performance metrics. Although the Frechet Inception Distance (FID) was common among all, metrics like the Inception Score (IS) and Perceptual Distance (PD) were not. Moreover, although the inference time for some models was specified, the hardware used differed, making it difficult to make any meaningful comparison between these times.

Chapter 3

Research Methodology

3.1 Problem Statement

The colorization of grayscale images is an ill-posed problem that can be solved using various techniques. Even among the deep-learning methods, there is a variety of approaches and architectures (CNNs, GANs, Vision Transformers, Diffusion models) that can perform well at this task. However, they vary widely in terms of accuracy, speed and scalability, making it difficult to determine the most suitable architecture for this problem. Therefore, we find it necessary to undergo a thorough comparison of these architectures to determine the one best suited for producing accurate, high-quality colorized images as no such comparison exists.

3.2 Hypothesis

We hypothesize that the Diffusion model is the best model for colorization in terms of accuracy and scalability. Their power to accurately learn the distribution of pixel values in an image has allowed them to produce high-quality results (accuracy) on large and complex datasets (scalability) in various Vision tasks. Colorization is no exception as the current state-of-the-art model is Palette (a diffusion-based model that considerably outperforms previous models in both qualitative and quantitative measures).

We also hypothesize that they are the worst model in terms of speed. Diffusion models generally have longer training and inference times than other models and thus we expect them to perform worse on these metrics.

3.3 Research Questions

1. Do diffusion models consistently score better on the FID, IS & PD quantitative metrics?

2. Do diffusion models scale better than other models (i.e., when the size of the training dataset increases, does the performance improve by a higher margin than in other models)?

3. Are the training times for diffusion models consistently higher?

4. Are the inference times for diffusion models consistently higher?

3.4 Methodology

3.4.1 Research Design

For this project, we will implement 3 classes of colorization models based on the GAN, Transformer and Diffusion architectures. The models within each class will be largely the same, but we will experiment with differently tuned hyperparameters to determine the best configuration for each class.

Once the best-performing models for each architecture have been determined, they will be compared on the previously mentioned quantitative measures in order to prove our hypothesis.

Our implementations will be predominantly based on the state-of-the art models of each architecture instead of making our own from scratch. However, one major change we will undergo is to scale these models down while still maintaining satisfactory levels of performance. The goal is only to compare these models at the task of colorization and striving for the state-of-the-art performance is unnecessary and infeasible as that requires enterprise-level computation power. Working on these smaller, less-trained models may also be beneficial as further credence may be given to our claim. Since, if diffusion still outperforms the other architectures even under these conditions, it could mean that this is the trend under all scales.

We will also prioritize training all models on the same hardware, so that comparisons on speed are meaningful, and making all models be of similar sizes, so that comparisons on accuracy are meaningful.

3.4.2 Methods

The implementation of this project will occur in 4 different phases. 3 representing each architecture and a final phase for comparing the architectures against each other.

Maximum Likelihood Estimation (MLE)

Agrawal [2021] describes MLE as the process of using data to find estimators for different parameters characterizing a distribution. This means that given a set of datapoints, MLE provides a way estimate the distribution that likely produced these datapoints. The estimated distribution can then be used to produce new data that can plausibly fit in with the old data without explicitly knowing the true distribution beforehand.

For colorization, the pixel values in an image are not wholly independent. They follow their own distribution and MLE can learn that distribution to ensure that the colors of any given pixel are consistent with the rest of the image.

We focus on the log of the likelihood because it is easier to calculate and ensures the results of our calculations do not tend to 0 or infinity [Taboga 2021]. By making the loss function of a neural network the negative log-likelihood and training to minimize it, we are attempting to make our distribution as close to the actual distribution as possible. This leads to the network learning an accurate distribution of the data that leads to good colorizations. The diffusion and transformer models to be implemented will make use of loss functions based on the negative log-likelihood.

$$\begin{aligned} L(\theta;\xi) = f(\xi;\theta) &= \prod_{i=1}^{n} fx\left(x_i;\mu,\sigma^2\right) \\ &= \prod_{i=1}^{n}\left(2\pi\sigma^2\right)^{-1/2}\exp\left(-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}\right) \\ &= \left(2\pi\sigma^2\right)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right) \end{aligned}$$

Figure 3.1: Maximum Likelihood Formula [Taboga 2021]

$$\begin{aligned} l(\theta;\xi) &= \ln[L(\theta;\xi)] \\ &= \ln\left[\left(2\pi\sigma^2\right)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)\right] \\ &= \ln\left[\left(2\pi\sigma^2\right)^{-n/2}\right] + \ln\left[\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2\right)\right] \\ &= -\frac{n}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 \\ &= -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 \end{aligned}$$

Figure 3.2: Log Likelihood Formula [Taboga 2021]

Diffusion

We will be implementing the Palette model [Saharia *et al.* 2022]. Although Palette is a unified architecture that can work on multiple image-to-image translation tasks (uncropping, colorization, inpainting, and JPEG restoration), we will only be using it for colorization.

The denoising model will be a U-Net with self-attention layers.

The source image will be the grayscale image and the denoising model will be trained to predict the full color image. After iteratively adding noise to the image we will train and perform inference using the algorithms in Figure 3.3. The gradient descent step is maximizing a weighted variational lower-bound on the likelihood. This is equivalent to maximizing the likelihood itself.

Algorithm 1 Training a denoising model f_θ

```
1: repeat
2:     $(\boldsymbol{x}, \boldsymbol{y}_0) \sim p(\boldsymbol{x}, \boldsymbol{y})$
3:     $\gamma \sim p(\gamma)$
4:     $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:     Take a gradient descent step on
           $\nabla_\theta \left\| f_\theta(\boldsymbol{x}, \sqrt{\gamma}\boldsymbol{y}_0 + \sqrt{1-\gamma}\boldsymbol{\epsilon}, \gamma) - \boldsymbol{\epsilon} \right\|_p^p$
6: until converged
```

Algorithm 2 Inference in T iterative refinement steps

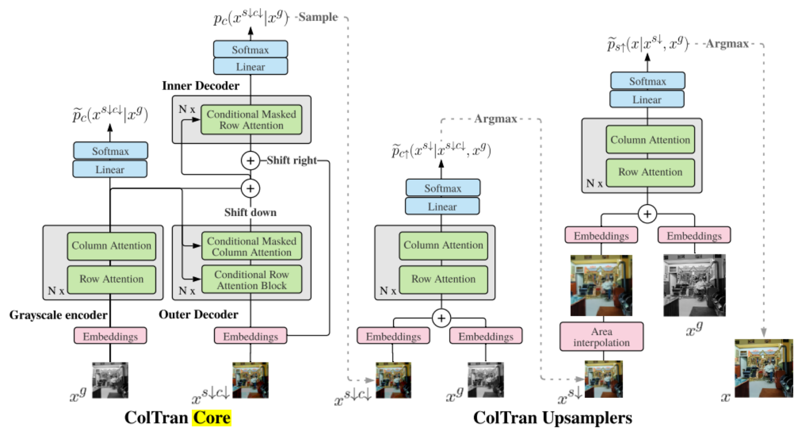
```
1: $\boldsymbol{y}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \dots, 1$ do
3:     $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
4:     $\boldsymbol{y}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{y}_t - \frac{1-\alpha_t}{\sqrt{1-\gamma_t}} f_\theta(\boldsymbol{x}, \boldsymbol{y}_t, \gamma_t)\right) + \sqrt{1-\alpha_t}\,\boldsymbol{z}$
5: end for
6: return $\boldsymbol{y}_0$
```

Figure 3.3: Palette Algorithms [Saharia *et al.* 2022]

Transformer

We will be implementing the ColTran model as depicted in Figure 3.4 and as described in the Related Work section [Kumar *et al.* 2021]. The Transformer components will be conditional according to Figure 3.5.

We will employ the objective function in Figure 3.6, where the probability at each step is given by how likely it was to produce the given image given the conditioned information. E.g., p_c is the probability of producing a particular coarse colorization given the original grayscale image. $\tilde{p}_c$ is used for the auxiliary parallel model with the goal of forcing the main to learn the color structure of an image early to produce better results. λ is a hyperparameter that determines the contribution of the auxiliary parallel model.



$p_c(x^{s\downarrow c\downarrow}|x^g)$ Sample
Softmax
Linear
Inner Decoder
N x
Conditional Masked Row Attention
Shift right
Shift down
$\widetilde{p}_c(x^{s\downarrow c\downarrow}|x^g)$
Softmax
Linear
Column Attention
Row Attention
N x
Conditional Masked Column Attention
Conditional Row Attention Block
N x
Grayscale encoder
Outer Decoder
Embeddings
Embeddings
x^g
$x^{s\downarrow c\downarrow}$
ColTran Core
Argmax
$\widetilde{p}_{c\uparrow}(x^{s\downarrow}|x^{s\downarrow c\downarrow}, x^g)$
Softmax
Linear
Column Attention
Row Attention
N x
Embeddings
Embeddings
$x^{s\downarrow c\downarrow}$
x^g
$\widetilde{p}_{s\uparrow}(x|x^{s\downarrow}, x^g)$ Argmax
Softmax
Linear
Column Attention
Row Attention
N x
Embeddings
Embeddings
Area interpolation
$x^{s\downarrow}$
x^g
x
ColTran Upsamplers

Figure 3.4: Coltran Structure [Kumar *et al.* 2021]

Component	Unconditional	Conditional
Self-Attention	$\mathbf{y} = \text{Softmax}(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{D}})\mathbf{v}$	$\mathbf{y} = \text{Softmax}(\frac{\mathbf{q}_c\mathbf{k}_c^\top}{\sqrt{D}})\mathbf{v}_c$ where $\forall \mathbf{z} = \mathbf{k}, \mathbf{q}, \mathbf{v}$ $\mathbf{z}_c = (\mathbf{c}U_s^z) \odot \mathbf{z} + (\mathbf{c}U_b^z)$
MLP	$\mathbf{y} = \text{ReLU}(\mathbf{x}U_1 + \mathbf{b}_1)U_2 + \mathbf{b}_2$	$\mathbf{h} = \text{ReLU}(\mathbf{x}U_1 + \mathbf{b}_1)U_2 + \mathbf{b}_2$ $\mathbf{y} = (\mathbf{c}U_s^f) \odot \mathbf{h} + (\mathbf{c}U_b^f)$
Layer Norm	$\mathbf{y} = \beta\,\text{Norm}(\mathbf{x}) + \gamma$	$\mathbf{y} = \beta_c\,\text{Norm}(\mathbf{x}) + \gamma_c$ where $\forall \mu = \beta_c, \gamma_c$ $\mathbf{c} \in \mathbb{R}^{H \times W \times D} \rightarrow \hat{\mathbf{c}} \in \mathbb{R}^{HW \times D}$ $\mu = (\mathbf{u} \cdot \hat{\mathbf{c}})U_d^\mu \quad \mathbf{u} \in \mathbb{R}^{HW}$

Figure 3.5: Conditional Transformer Components [Kumar *et al.* 2021]

$$\mathcal{L} = (1 - \lambda)\log p_c + \lambda \log \widetilde{p}_c + \log \widetilde{p}_{c\uparrow} + \log \widetilde{p}_{s\uparrow}$$

Figure 3.6: ColTran Objective Function [Kumar *et al.* 2021]

GAN

Our GAN implementation will be based on Pix2Pix [Isola *et al.* 2017], a cGAN variant designed for general image-to-image tasks where the generator is a U-Net and the discriminator is a convolutional PatchGAN Classifier. A PatchGAN discriminator attempts to classify each NxN patch in an image as real or fake and averages all responses to find the final output.

For the objective we combine the objective of a conditional GAN (1) with an L1 (Least Absolute Deviations) loss to obtain the final objective G*. Where the discriminator maximizes the cGAN loss, the generator minimizes said maximum and the L1 loss is tuned with a hyperparameter λ.

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))],$$

Figure 3.7: cGAN Objective Function [Isola *et al.* 2017]

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

Figure 3.8: L1 Loss [Isola *et al.* 2017]

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G).$$

Figure 3.9: PixColor Objective Function [Isola *et al.* 2017]

3.4.3 Dataset

The dataset to be used for this project will be ImageNet [Russakovsky *et al.* 2015]. It contains a large number of diverse images and is the standard for Vision tasks.

A small subset of approximately 100000 images (to be varied in the final phase) out of the 1.2 million ImageNet training set will be used for training. 10000 and 5000 images from the ImageNet validation and test sets respectively will be used to tune and, subsequently, evaluate our models. As these images will be in a variety of resolutions, they will all be scaled to a resolution of 256x256 during pre-processing.

3.4.4 Implementation Details

Python will be predominantly used as there are many deep-learning libraries that exist that can implement the various neural network architectures. Moreover, there are many libraries for statistical analysis that will be useful towards the end of the research when the models are being compared.

For the implementation of our models, we will make use of the TensorFlow library specifically. It has the capacity to implement all the necessary architectures and since some of the state-of-the-art models are in this library (e.g., ColTran), it will be easier to adapt and tailor those models to suit the needs of this research project.

3.4.5 Performance Evaluation

For all 5000 test images we will calculate and evaluate the average FID, IS and PD produced by each trained model. Each of these metrics will be calculated over an interval of training steps to determine how each model improves with the number of training steps.

For our final three models we will compare these metrics in addition to the training and inference times and the training set size to prove our hypotheses.

3.4.6 Limitations

It is possible that computational resources may be a limiting factor. Although we wish to create small-medium scale models, there might be difficulty in trying to scale the large models down while maintaining good levels of performance ($>=70\%$ of the original).

Chapter 4

Work Schedule

Table 4.1: Work Schedule

Number	Task Description	Start Date	End Date	Time
	Setup	**03/07/2023**	**10/07/2023**	**1 Week**
1	Download and pre-process ImageNet images			
2	Implement FID, IS & PD calculations			
	Implement cGAN model	**10/07/2023**	**31/07/2023**	**3 Weeks**
3	Implement Generator			
4	Implement PatchGAN Classifier			
5	Create variants with different hyperparameters and evaluate them			
6	Find ideal set of hyperparameters for final cGAN model			
	Implement Diffusion model	**31/07/2023**	**28/08/2023**	**4 Weeks**
7	Implement forward diffusion process			
8	Implement denoising model and reverse diffusion process			
9	Implement inference process			
10	Create variants with different hyperparameters and evaluate them			
11	Find ideal set of hyperparameters for final Diffusion model			
	Implement Transformer model	**28/08/2023**	**02/10/2023**	**5 Weeks**
12	Implement Coltran Core			
13	Implement Coltran Color Upsampler			
14	Implement Coltran Spatial Upsampler			
15	Implement Parallel Auxiliary Model			
16	Create variants with different hyperparameters and evaluate them			
17	Find ideal set of hyperparameters for final Transformer model			
18	**Evaluate final 3 models when trained on increasing sizes of data**	**02/10/2023**	**09/10/2023**	**1 Week**
19	**Analyse final results and finish report**	**09/10/2023**	**23/10/2023**	**2 Weeks**
*20	**Write report**	**03/07/2023**	**23/10/2023**	**16 Weeks**

Chapter 5

Conclusion

Although image colorization is an old problem, there are still many new methods being proposed that continually push the state-of-the-art (the latest of which being Diffusion colorization models). Besides having tangible scientific use, they can also make appealing and diverse colorizations for aesthetic purposes. This is impressive considering that 1-dimensional data is being used to produce 3-dimensional data.

There is a lack of thorough review/comparison of different colorization models to determine which architectures are better or worse in particular aspects. Therefore, we propose a method to reconcile this, with the underlying assumption that Diffusion models produce the best results. This is from the fact that the colours in an image are not wholly independent of each other and form a predictable distribution, and diffusion models have the ability to learn distributions well.

References

[Agrawal 2021] Naman Agrawal. *Maximum Likelihood Estimation -A Comprehensive Guide*. https://www.analyticsvidhya.com/blog/2021/09/maximum-likelihood-estimation-a-comprehensive-guide/, 2021. Accessed: 2023-05-14.

[Ardizzone *et al.* 2019] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[Brownlee 2019] Jason Brownlee. How to identify and diagnose gan failure modes. *Internet: https://machinelearningmastery. com/practical-guide-to-gan-failuremodes*, 2019.

[Cao *et al.* 2017] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 10*, pages 151–166. Springer, 2017.

[Cheng *et al.* 2015] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE international conference on computer vision*, pages 415–423, 2015.

[Code 2022] Papers With Code. *Colorization on ImageNet val*. https://paperswithcode.com/sota/colorization-on-imagenet-val, 2022. Accessed: 2023-05-14.

[Goodfellow *et al.* 2020] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[Guadarrama *et al.* 2017] Sergio Guadarrama, Ryan Dahl, David Bieber, Mohammad Norouzi, Jonathon Shlens, and Kevin Murphy. Pixcolor: Pixel recursive colorization. *arXiv preprint arXiv:1705.07208*, 2017.

[IBM nd] IBM. *Convolutional Neural Networks*. https://www.ibm.com/topics/convolutional-neural-networks, n.d. Accessed: 2023-05-14.

[Isola *et al.* 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the*

IEEE conference on computer vision and pattern recognition, pages 1125–1134, 2017.

[Khan *et al.* 2022] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[Kumar *et al.* 2021] Manoj Kumar, Dirk Weissenborn, and Nal Kalchbrenner. Colorization transformer. *arXiv preprint arXiv:2102.04432*, 2021.

[Li *et al.* 2020] Bo Li, Yu-Kun Lai, and Paul L Rosin. A review of image colourisation. *Handbook Of Pattern Recognition And Computer Vision*, pages 139–157, 2020.

[Peerspace 2015] Peerspace. *When was Color Photography Invented*. https://www.peerspace.com/resources/when-was-color-photography-invented/#:~:text=Popularization&text=It%20wasn%27t%20an%20immediate,the%201960s%20to%20the%201970s, 2015. Accessed: 2023-05-14.

[Russakovsky *et al.* 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[Saharia *et al.* 2022] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

[Sohl-Dickstein *et al.* 2015] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[Su *et al.* 2020] Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7968–7977, 2020.

[Taboga 2021] Marco Taboga. *Log-likelihood*. https://www.statlect.com/glossary/log-likelihood, 2021. Accessed: 2023-05-14.

[Vaswani *et al.* 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.* 2022] Yi Wang, Menghan Xia, Lu Qi, Jing Shao, and Yu Qiao. Palgan: Image colorization with palette generative adversarial networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XV*, pages 271–288. Springer, 2022.

[Weng 2021] Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021.

[Zhang *et al.* 2016] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 649–666. Springer, 2016.