Evaluation of Image Colourization Approaches

Filip Appelgren **Jakob Berggren** **Erik Båvenstrand** **Oskar Hahr**

Abstract

This paper delves into the modern ways of adding colour to monochrome - black-and-white - images, through the use of deep learning. We take inspiration from previous works to implement a U-Net architecture that attempts to colourize 64x64 images taken from the Imagenet dataset. The network is used to implement two different methods of colourization, one through regression and the other through classification. The training was done on a dataset of 50000 images for the regression, and 30000 for classification. The results indicate successful colourizations using both methods with the U-net architecture. The way of classification shows more vibrant colours than regression, but also assesses bolder colours which can lead to strange results. Improvements to the results could possibly be achieved through a variety of adjustments, such as a larger dataset, a more advanced network as well as training the network training.

1 Introduction

Colourization of greyscale images or movies is not a new concept in itself; it has been around since at least the start of the 18th century, when filmmaker Georges Méliès famously hand-coloured some of his movies, with the help of a studio and over 200 people [1]. Today, colourization is not done by hand-painting directly on film, but can instead be performed through the use of neural networks that rely on object and texture detection from greyscale images. The goal of such a network is not necessarily to return the original colour that would have been visible in the ground-truth image, but rather to return a colour that may look plausible to a human observer. It has to do with the fact that there is rarely an explicit mapping between colour and texture applicable to all images.

The goal of this project is to train two identical networks, with different output layer and loss function approaches. One network will be trained by directly mapping a greyscale pixel to the corresponding colour trough regression while the other network will be trained by classifying the colour of a pixel to a set number of possible colours. The networks will be trained on roughly the same amount of data, and once finalized, will be evaluated by the subjective quality of the colourization. The idea is that this will give us insight into how well a regression-based network compares to a classification-based network in the task of image colourization.

2 Related work

In the past few years, there have been numerous papers published on the topic of image colourization through the use of deep learning. *Colorful Image Colorization* by Zhang et al. [7], describes the process to divide the colourspace into evenly sized buckets that transform the image colourization problem from a regression problem to a classification problem.

The primary source of inspiration for the network architecture in this paper is *U-Net: Convolutional Networks for Biomedical Image Segmentation* [4] by Ronneberger et al. They proposed a

[1] `https://wfpp.columbia.edu/essay/french-film-colorists/`

modified autoencoder network with sparse connections between same-sized layers that perform well on feature extraction tasks.

Furthermore, a paper from Stanford University, *ColorUNet: A Convolutional Classification Approach to Colorization*, describes an approach to this problem via a deep convolutional neural network using classification. One of their most significant findings is the fact that the colour bucket plays a significant role in the accuracy of the model, and that picking suitable buckets for the training data is essential [1].

3 Background

3.1 Tiny-Imagenet

ImageNet is a large-scale hierarchical image database hosted by the vision lab at Stanford University. It is comprised of hundreds of thousands of images displaying each noun currently available in WordNet [2]. The resolution of the images is 256x256 pixels, which is considerably higher than the CIFAR-10 dataset. There are versions of ImageNet referred to as Tiny-Imagenet, that are already down-scaled to the resolutions 64x64 and 32x32 [6]. In this project, a 64x64 version of ImageNet has been used.

3.2 Network Architecture

3.2.1 Cross-channel Encoder

The Cross-channel Encoder is a neural network architecture proposed in *Colorful Image Colorization* by Zhang et al. [7]. The network takes a greyscale image as input in the form of 256x256x1 and outputs two colour channels A & B in the form of 224x224x2.

The network shares a resemblance with an Auto-encoder, in that it constricts the input down to a smaller size through several convolutional neural network layers. After which the network enlarges the input back to the desired output size through a similar process of expansion. The network is implemented using stacked convolutional layers with ReLu activation, and downsampling is performed by a 2x2 stride, on the last convolutional layer of each group.

3.2.2 U-Net

The U-Net was developed for biomedical image segmentation in 2015 [4] by Ronneberger et al. This implementation also follows the typical layout of an autoencoder network with a constriction and an expansion of data. The network is implemented with stacked convolutional layers with ReLU activation, grouped with a max-pooling layer with a stride of 2 that handles the downsampling. The expansion of the encoded data is handled by repeated upsampling combined with stacked convolutional layers.

The big difference between a U-Net and an autoencoder is that for each stacked convolutional layer group in the constriction phase, the output is sent both to the downsampled layer below it, but also transferred across the network to the corresponding layer in the expansion phase. This results in that additional extracted features available in each layer of the constriction phase are being used as input to the matching layer in the expansion phase. During the expansion phase, the network will then combine information from the corresponding layer in the constriction phase with the information from the layer below as it expands. This concatenation creates a better prediction by using all data available at that resolution size. Further, it is important to mention that dropout and batch normalization are utilized throughout the entire network to prevent overfitting.

3.3 Image Representation

The Tiny-ImageNet dataset represents images in the RGB colour space. In the subject of image colourization, it is very common to use the CIELAB colour space [1, 7], which is composed of the L channel which is the lightness where 0 indicates black and 100 indicates white. The two other channels A and B contain information about the colour of the image. This makes it easier to divide

the image into inputs (L channel) and targets (AB channels) for training. It has been shown that the CIELAB colour space is advantageous to use for colourization purposes [3].

3.4 Colourization as a Regression Problem

A relatively straight forward approach to the image colourization problem is with regression. Each image is converted to the CIELAB colour space, and the network is trained using the greyscale channel L as input. The network then outputs a prediction of the colour channels A & B. This type of network can be trained using a loss function such as Mean Squared Error. This approach is less demanding in terms of implementation compared to some other methods. However, there are drawbacks due to the nature of the loss function solely being based around the distance between the prediction value and the ground truth; these models typically favour more conservative guesses [1, 7]. The result of this is that the colourized images tend to have "washed out" colours similar to that of "sepia".

3.5 Colourization as a Classification Problem

When considering the drawbacks of image colourization with regression, there is the option to transform the problem to a classification problem [1, 7]. Image colourization by classification enables several methods to be applied that would steer the model away from picking "washed-out" colours.

The core idea of this approach is to divide the colour space into a finite number of buckets, thus creating discrete steps in colour picking. Zhang et al. describe a method to weight the colour buckets based on the frequency of the colour in the training dataset. By assigning a higher weight to uncommon colours, the network is more prone to produce colourful images. A post-processing step is added to control the intensity of the colours referred to as the temperature. More about the effect of this variable will be discussed in sections 4.2.1 and 5.4.

4 Methodology

4.1 Network Architecture

We initially constructed a network with a similar structure to that which was proposed by Zhang et al. This network ended up containing around 33 million parameters. A network of this size would require much time, resources, and a large dataset to properly train. After some experimentation by training the network, the decision was made to instead go with a smaller network model.

We decided to implement and perform the experiments on a downscaled version of the U-Net. Compared the Cross-channel Encoder described by Zhang et al. the U-Net has approximately 3.5 million trainable parameters, which is more suitable for our constraints in time and resources. The exact architecture we used during the experiments is detailed in table 1.

4.2 Two approaches to the same problem

Our implementation of the regression-based image colourization is mostly similar to that which is described in section 3.4. The network's input layer consists of a 64x64x1 input channel which represents the greyscale version of the image. The output of the network is a 64x64x2 representation of the colour channels A & B, according to CIELAB colour representation. We then used Mean squared error to calculate the loss between each pixel value from our predicted image to the ground truth image.

Our classification based image colourization approach was primarily inspired by *Colorful Image Colorization* by Zhang et al. [7]. We divided our colour space into 313 different buckets, where each corresponds to a particular colour. Furthermore, this allowed us to assign a weight to each colour bucket, based on how often that particular bucket appears in the training data. The weights of each bucket thereby correspond to the rarity of that colour in the training dataset. An image representation of the colour buckets can be found in figure 1a. We had to modify the last layer of the network to output 64x64x313 with a softmax activation function. This means that each pixel will have a probability distribution of possible buckets from which the colour will be picked. The weight w of

Layer	X	C	S	Dr	BN	Co
Input	64x64	1	-	-	-	-
Conv1_1	64x64	32	1	0.2	-	-
Conv1_2	64x64	32	1	-	-	-
MaxPool_1	32x32	32	-	-	Yes	-
Conv2_1	32x32	64	1	0.2	-	-
Conv2_2	32x32	64	1	-	-	-
MaxPool_2	16x16	64	-	-	Yes	-
Conv3_1	16x16	128	1	0.2	-	-
Conv3_2	16x16	128	1	-	-	-
MaxPool_3	8x8	128	-	-	Yes	-
Conv4_1	8x8	256	1	0.2	-	-
Conv4_2	8x8	256	1	-	Yes	-
UpSample	16x16	256	2	-	-	-
Concatenate	16x16	384	-	-	-	Conv3_2
Conv5_1	16x16	128	1	0.2	-	-
Conv5_2	16x16	128	1	-	Yes	-
UpSample	32x32	128	2	-	-	-
Concatenate	32x32	192	-	-	-	Conv2_2
Conv6_1	32x32	64	1	0.2	-	-
Conv6_2	32x32	64	1	-	Yes	-
UpSample	64x64	64	2	-	-	-
Concatenate	64x64	96	-	-	-	Conv1_2
Conv7_1	64x64	32	1	-	-	-
Conv7_2	64x64	32	1	-	-	-
Output	64x64	2 / 313	1	-	-	-

Table 1: U-Net network architecture used for both regression and classification.
X dimensions of output; **C** number of channels of output; **S** stride; **Dr** dropout after layer; **BN** whether BatchNormalization layer was used after layer; **Co** layer concatenated with the UpSample layer on the row before it

each colour bucket is retrieved using the following formula:

$$w = ((1-\gamma)\mathbf{p} + \frac{\gamma}{Q})^{-\alpha}.$$

Where **p** is the probability of a specific bucket, Q is the number of colour buckets, and α & γ are variables that control how much the final probability will be reweighed.

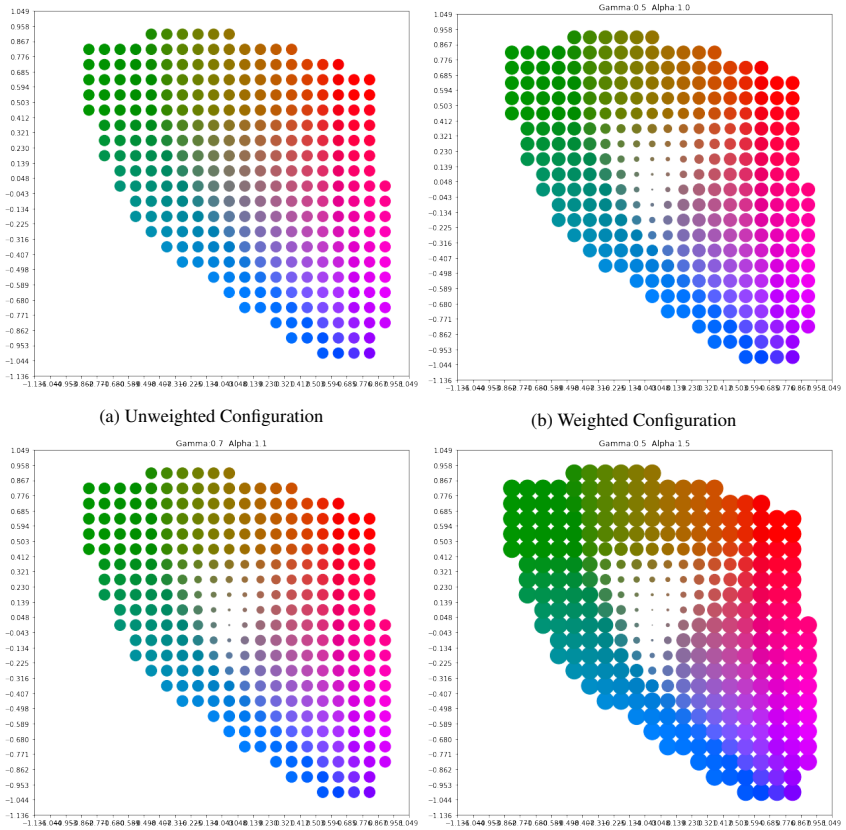
4.2.1 Annealed Mean

Once our classifier has output a probability vector of possible buckets to pick a colour from, the decision has to be made on how to decide which colour to pick. One technique of doing this would be to take the mode (or max argument) of the probability distribution. This can lead to very extreme colour choices, for example, if a shade pink has the largest probability, but there exists a range of 3 different greens that together have a higher probability, the model would still pick pink. A way of solving this is to take the mean colour based on the probability distribution; however, doing this is largely equivalent to how the regression model chooses colours.

To solve this we used what Zhang et al. refers to as taking the Annealed Mean of the probability distribution [7]:

$$f_T(\mathbf{Z}) = \frac{exp(log(\mathbf{z})/T)}{\sum_q exp(log(\mathbf{z}_q)/T)}.$$

This introduces the temperature term T. As $T \to 0$ the model will pick colours based on the mode of the probability distribution **Z** leading to more extreme colour choices. As $T \to 1$ the model will move towards taking the mean colour of **Z** leading to more washed-out colours. Tuning the temperature factor will hopefully allow us to get the best of both worlds.



(a) Unweighted Configuration

(b) Weighted Configuration

(c) Weighted Configuration

(d) Weighted Configuration

Figure 1: Colour bucket with Different Weights

4.3 Data

We decided to use a 50000 sample subset of the Tiny-ImageNet 64x64 dataset for training the regression-based networks. The considerably smaller dataset was chosen due to limited computational resources and time constraints. Unfortunately, the classification target images are of size 64x64x313 and took up an excess of 200GB in space. This lead to the decision to reduce the amount of training data, which means that the training set of the regression model contained 50000 images while the training set for the classification model contained 30000 images.

When dealing with large amounts of training data, it is common that the RAM is a bottleneck during training. To counteract this an input pipeline was built using the Dataset API provided by TensorFlow. The dataset was divided into files containing 128 serialized images, to speed up reading the data. This improved the training speed of the model considerably.

5 Experiments

The selection of the testing images was conducted by randomly sampling a test set of images never before seen by any of the models.

5.1 Initial experiments

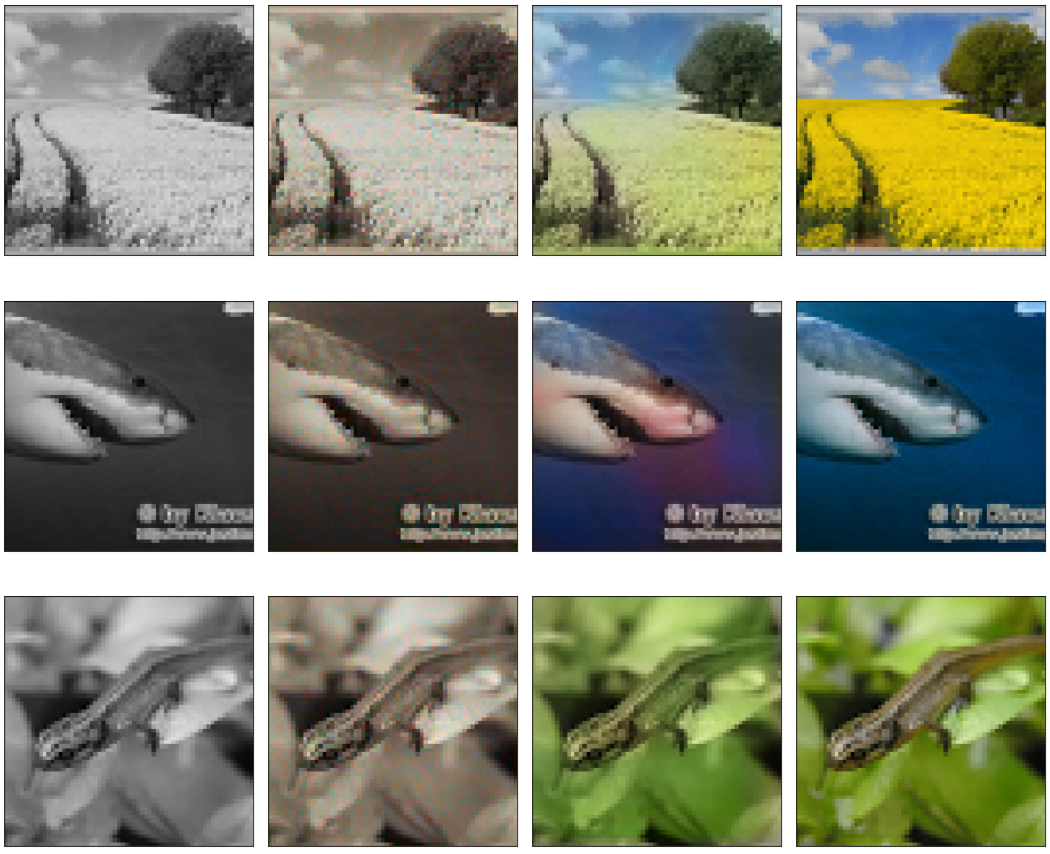
In the initial experiments, we trained a smaller autoencoder network with regression on the CIFAR-10 dataset. These experiments were conducted to among other things to test functions related to data handling such as:

- RGB to CIELAB conversion and vice versa.

Figure 2: CIFAR-10: Greyscale, Prediction, Ground Truth

- Pre-loading data into the Tensorflow Record format for faster training and processing.

We also trained a small version of the final U-Net on 10 CIFAR images to see if the network converged. After confirming that these functions were working as intended, we quickly moved on to building models using the Tiny-ImageNet dataset for training & validation and the full-sized U-Net. The results can be seen in figure 2.



(a) Greyscale (b) SGD Prediction (c) Adam Prediction (d) Ground Truth

Figure 3: Comparison of SGD and Adam Optimization Function with Regression

5.2 Regression: Choice of Optimization Function

In this experiment, we compare the performance of two different optimization functions - Stochastic gradient descent (SGD) and Adaptive Moment Estimation (Adam). SGD is an optimizer similar

to regular gradient descent but is much more efficient by performing parameter updates for each training example instead of the whole dataset. Adam is a more modern optimizer that has been proven to work well for deep learning [5].

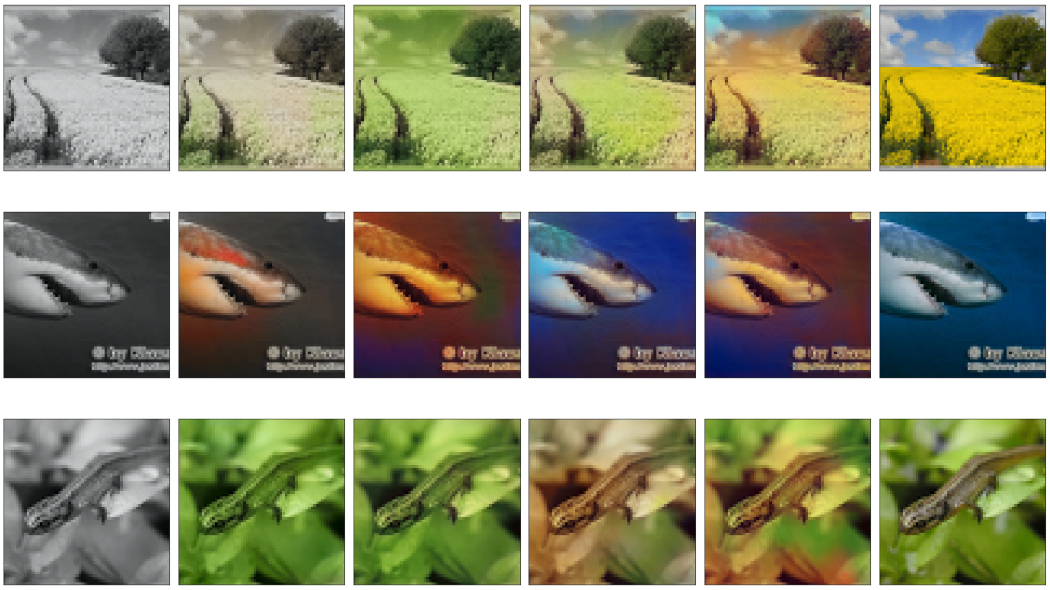
The experiment was conducted by training two different networks sharing the same design, except for the optimization function, where one network utilized SGD, and the other Adam. The training was conducted on 50000 samples and using a batch size of 64, for 55 epochs.

The result of the experiment can be seen in figure 3. By qualitative comparison of randomly selected images from the test set, we can conclude that Adam outperforms SGD in this experiment.

5.3 Bucketed Colour Weight Comparison

To evaluate how the weight of bucketed colours affect the result, we designed an experiment where we trained four models with four different colour bucket weight configurations, that are visualized in figure 1. There is one that is unweighted/uniformly weighted, and three different weight configurations with an increasing discrepancy between common and uncommon buckets.

The results of the experiment are shown in figure 4 which displays three pictures from the test set. From these images, we can observe that the colouring trends towards less defensive colouring as we let the weights go towards more extreme values. Subjectively we find that an alpha value of 1.1 gives the best results in this experiment.



(a) Greyscale (b) Unweighted (c) alpha=1.0 (d) alpha=1.1 (e) alpha=1.5 (f) Ground Truth

Figure 4: Comparison of Color Weight Configurations in Classification

5.4 Temperature

We experimented with the temperature variable discussed in section 4.2.1 by linearly incrementing it from $T \to 0$ to $T \to 1$. In figure 5, the effect of the temperature variable can be seen. At very low temperatures, the colour separation is more distinct, just as anticipated. At high temperatures, the colours start to become "washed-out". A good value for the temperature seems to be $0.23 < T < 0.40$ just as Zhang et al. found [7]. The model used in this experiment was the one from figure 4d but trained for a total of 50 epochs.

Figure 5: Effect of the Temperature Variable

5.5 Regression vs Classification

We compared the best regression model with the best classification model as a final experiment. The images in figure 6 display the results. While regression produces the most plausible colours, they are rather "washed-out" as seen in the previous experiments. Classification, on the other hand, produced more vibrant results but with a greater deal of miscolouring. An example of this can be seen in the dog image where the water and sky is a shade of yellow.

6 Discussion

6.1 Optimization Functions

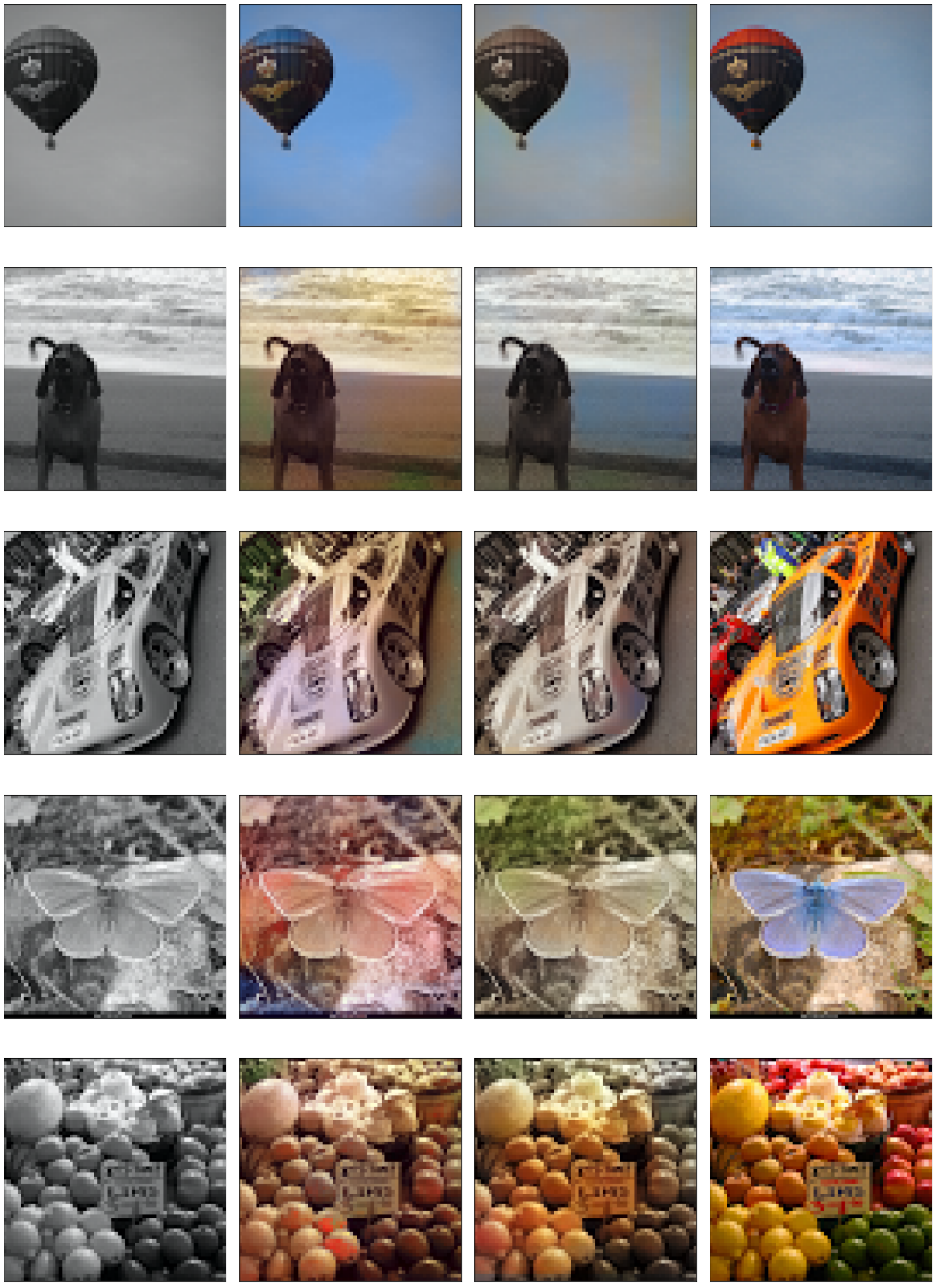
Based on the results we have seen from the optimizer experiment, we can conclude that the choice of optimizer has a significant impact on how long it takes to train a model and what the final results will be. The SGD model ended up performing a lot worse than the ADAM optimizer when trained for the same amount of epochs and on the same data. If the SGD model had been trained for a more extended period of time, it is possible it could have produced similar results. Other factors that could have impacted this is the choice of the network model, choice of the loss function, dataset, amongst other things. Nevertheless, on this type of problem, the ADAM optimizer seemed vastly superior with our model.

6.2 Training Data

We trained our models with a relatively small dataset of 50000, and 30000, images respectively. These datasets also contained a vast range of different motives. This, in turn, leads to that our models (some more successful than others) were generally able to learn patterns such as grass, sea and the sky, but struggled with more specific patterns such as colours of objects or animals. If we had chosen to either include a larger dataset or to limit the dataset to for example landscape images, we could have potentially improved our results within those specific domains.

Another factor that may have impacted the results of our experiments is the chosen resolution. The previous works that inspired our models have typically used the full 256x256 resolution of Imagenet. Our smaller resolution of 64x64 may have made it harder for the network to properly learn distinct patterns between objects in images, for example.

However, using a larger dataset or larger image resolution would have required more computational resources that were not available to us. Despite this, given the computational and time limitations, we are still satisfied with the results that our models were able to produce.



(a) Greyscale (b) Classification (c) Regression (d) Ground Truth

Figure 6: Comparison of Classification and Regression on Test Images

6.3 Network Architecture

Despite the somewhat successful colourization models, we think that the limited size of our U-Net implementation could have hindered further improvement given a more extended training duration. Had we instead opted to implement the Cross-channel Encoder or the full-sized U-net, the results

could have possibly turned out better. However, these larger networks would depend on both having access to more computational resources and a larger dataset in order to be able to produce any meaningful results.

6.4 Tuning Variables

Our experiments on tuning temperature values & weight calculation show that tuning these hyperparameters can significantly impact the final results of a given classification model. To further improve the results of the model, a more thorough grid search could be conducted in order to fine-tune the hyperparameters further.

References

[1] Vincent Billaut, Matthieu Rochemonteix, and Marc Thibault. Colorunet: A convolutional classifcation approach to colorization. Master's thesis, Stanford University, 6 2018.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Shreyank N Gowda and Chun Yuan. Colornet: Investigating the importance of color spaces for image classification. In *Asian Conference on Computer Vision*, pages 581–596. Springer, 2018.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[5] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[6] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Downsampled imagenet 64x64.

[7] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.