Coloring the Past - Image Colorization

Fabian Aguilar Gomez
The University of Texas at San Antonio
1 UTSA Circle, San Antonio, TX 78249
fcy527@my.utsa.edu

Noor Alaskari
The University of Texas at San Antonio
1 UTSA Circle, San Antonio, TX 78249
tpd972@my.utsa.edu

Ivan Rivera
The University of Texas at San Antonio
1 UTSA Circle, San Antonio, TX 78249
ofv948@my.utsa.edu

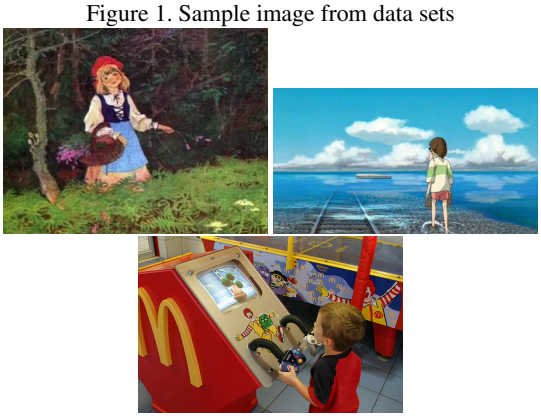
Abstract

Historic images allow us to see how the world was with a visual representation, but with no color, it's hard to fully connect with the images. This is important since images having color allow it to come to 'life' and allows the individual dissecting and analyzing the image to digest the image fully through color. As a group of three computer science seniors who have taken coursework such as Artificial Intelligence and Data Science we will implement a Convolutional Neural Network (CNN) to produce some color and build upon this simple model. Using Google Colaboratory and python libraries such as Tensorflow, Keras, Open CV, Skimage, NumPy, etc. this problem won't be impossible to solve. Our idea and tools are feasible since Google Colaboratory is a free source to create and edit code and also collaborate with our team members and allow for real-time collaboration and contains all needed libraries already installed. Success for us will look like at the minimum have some form of color in Black and White images (B&W) that we have trained.

1. Introduction

Image colorization has been popularized as of recently with the drive to attempt to colorize old images & video alike, most note ably black and white images. There have been many different neural network architectures that have emerged in the attempt to have the most accurate color representation for each input given, the image/video. The most popular *DeOldify*[1] was created by Jason Antic which used 3 separate models, primarily GAN models to create one of the most accurate image colorizations out there. Considering the scope of this class we settled on implementing the idea of image colorization using a CNN instead, as thus far we have far more knowledge of these types of networks. We intend to use multiple data sets as we would want to see how well the model interacts in general use, being different images. For now we have set our eyes on a couple particular datasets such as Art Images[4], Japanese Anime Scenes[7], and Flickr Image dataset[5] all of these datasets can be found in *Kaggle*. Each of the images is of different dimensions so we will resize the images to be that of size $224x224$ and having 3 channels corresponding to RGB. Our simple first model will be vanilla CNNs in which we will create the most basic network that has been well established for image colorization[9]. Building a successful basic model will allow for improvements & expansion of the model.

Figure 1. Sample image from data sets



2. Related Work

2.1. CNN

The project takes most of the influence from the paper *Colorful Image Colorization* which uses the simplest network architecture [9]. The network generates a Color AB

Figure 2. Basic CNN for image colorization

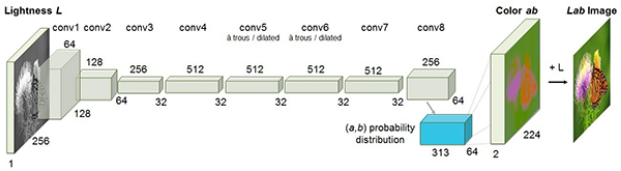
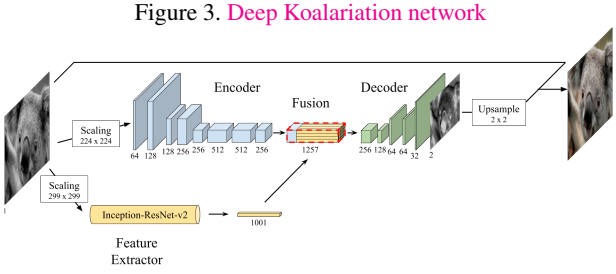


image which is the same dimensions as the input image that the model was trained on. This is accomplished by mapping the grayscale input from the image to quantized color value outputs [9]. Thus, when added with L image we will produce a colorized image, which attempts to be identical to the ground truth. The addition of these two images combined allows the trained image or images that are closely related to the training images to be colorized with pretty high accuracy and colors being in the correct place. The model can accomplish this by what is mentioned in the paper of treating the learning as a *cross-channel encoder* [9]. The issue with this architecture is that it does an extremely good job when colorizing the input image and images that are similar in retrospect to the training images. It makes sense as the images closely related to the training images it will output images with good accuracy in colorizing, this is why datasets such as *ImageNet* are beneficial and could provide better overall results. This simple model however does not do so great when it comes to other images that the model has not been trained or in short 'generalizing' the problem. This is because when attempting to colorize say a cartoon image the weights that the model has been trained on are that of 'real life' examples (e.g. a person, an event, or scenery). Thus, models have emerged to attempt to solve the problem and 'generalize' the solution of colorization.

2.2. CNN with ResNet-v2 and feature extraction

The paper *Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2* attempts to solve this issue. The model presented is one of the best since it takes

Figure 3. Deep Koalariation network



in any input regardless of their input size & aspect ratio, which is a huge improvement on the initial architecture that is discussed in the first paper [3]. Continuing, this is a powerful addition since being able to take images from any input size will allow for broader dataset training since during the development of the model there won't be hassles to predetermine the input size. The idea is that the CNN model is combined with the *Inception-ResNet-v2* that has been already pre-trained on the *ImageNet* dataset. This part acts as the feature extractor part and extracts layers from the image (e.g. mid & high) which are later added to the fusion layer. Lastly, the model with the features added the decoder can estimate what the output should look like [3]. This model has been proven to be one of the best CNN architectures when it comes to image colorization as it also improves on the findings from the first paper. This is due to this architecture being able to colorize images better than the model has not been trained on and allow for a more rich color.

2.3. Chroma GAN

Lastly, this problem has not only been attempted with CNNs. Generative Adversarial Networks (GAN) have been used to and have produced some huge improvement. Chroma GAN is something new that has popped up last year, 2020. In this network, it combines a discriminator & a generator which attempts to infer the chromaticity of a grayscale input. Here the generator is the result of two subnetworks which are divided into three stages that share modules in between [8]. The model is trained to learn to colorize the image by using a perceptual and semantic understanding of color & class distributions. The discriminator architecture uses a *PatchGAN* architecture, which deals with penalizing only at the scale of local image patches[6]. This model had modified if not changed the wheel completely on the original established CNN model to colorize the image. To no surprise however with the model swapping to a GAN instead of a CNN and creating a more articulate generator & discriminator networks the results speak for themselves as the images colorized that have not been trained appear sharper and more 'colorful' for a lack of better word. This is due to the model being trained on what they mention as **legacy black and white photographs** that have chrominance information is removed [8]. Image col-

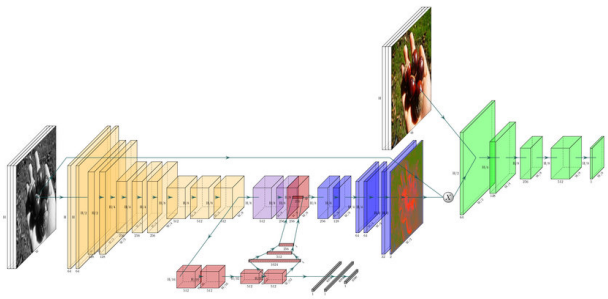


Figure 4. ChromaGAN architecture

orization models build upon the main model which was established in the first paper. The *Deep Koalariation Network*

built upon by adding the *Image-ResNet-v2* and also involving convolving up. Whilst the *ChromaGAN* architecture redesigns the model by implementing the idea using a GAN model.

3. Approach

Create the vanilla network and build our model from there since starting from the ground up would basically be extremely difficult.

3.1. Loading Data

The image needs to be cleaned and prepared for the model since we don't want any inconsistencies or have an image cause a change in the input. The data loaded will be RGB images of shape $256x256$ if the images do not match these dimensions a reshape of the image will take place. Thus, the final shape of an image from the training dataset will be $224x224x3$. The next step will be to convert the images from RGB values to Lab color space. For this specific color space **L** is the lightness, **a** & **b** are the color spectra green-red & blue yellow respectively. After successful loading and cleaning of the training data, we want to separate the *L* value and *ab* values from the image and train the model to predict the *ab* values.

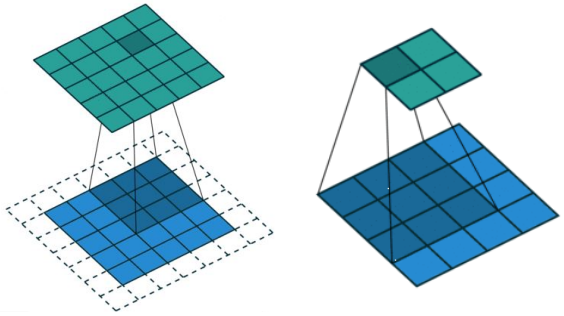
3.2. Initial setup

For our approach, we want to implement the original CNN that will colorize basic images as stated in the first paper [9]. We want to start with the foundation since if things go south or do not work well we will have a backup plan that will at least produce some results. We could also modify the original network by adding more layers and changing the inputs and such.

3.3. Layers

Our CNN will contain *Conv2D* layers which will take care of spatial convolution over the images we provided to the model. Additionally, we will use *UpSampling2D* layers dealing doubling the dimensions of the current input it gets from the previous *Conv2D* layers [9].

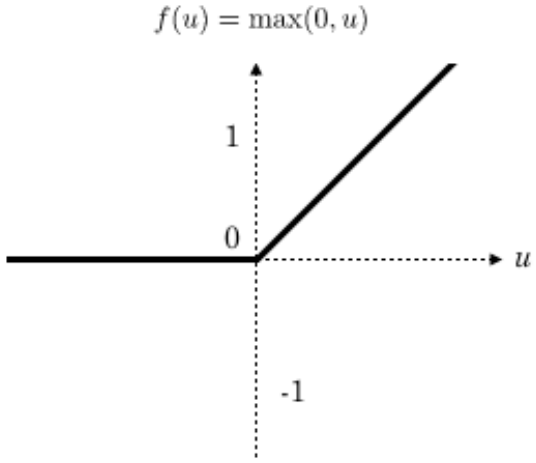
Figure 5. Conv2D and UpSampling2D



3.4. Padding and Activation functions

The padding used for the Conv2D layers will be *same*, meaning the padding will be enough to offset the kernel size which will create the output dim same as the input. Stride of 2 is used in the Conv2D layer in order to be able to decrement the *width* and *height* by a ratio of 0.5, or half. The activation function used for the layers will be *ReLu*, as this activation function tends to be best for general usage and implementation of CNNs. These parameters for the layers will serve as the 'base' to attempt to get the network to work before moving on to modifications of the network.

Figure 6. ReLu activation function



3.5. Proposed Approach

We want to attempt to change the original network by messing with it (e.g. removing/adding layers, fine-tuning parameters)[3]. We also want to attempt to mix different networks (e.g. append another network to the result of the other). We wish to play around with the models and attempt to see if we can improve on the networks. If we can mix two networks we think it should produce even better results considering the input is going through two different models. This differs from the ones that have been mentioned since the mixing of two networks is not mentioned by the papers. If this does not work we at least want to modify the initial network or the *Deep Koalariation network* to the point where we have drastically changed it and have done something different from them, maybe create a deep convectional network as opposed to just a simple CNN.

3.6. Architecture Modification

After the initial base vanilla model shows promising results we want to move on to modifying it and increasing it by implementing more layers and upsampling layers. Our model plan is to create a model with additional layers and deepen the network by using more Conv2D functions and

be able to upsample the input as mentioned using UpSample2D layers. We expect the model to be deeper than the original model proposed in the first paper with the intent that a deeper network will produce better results as it will have to go through more layers.

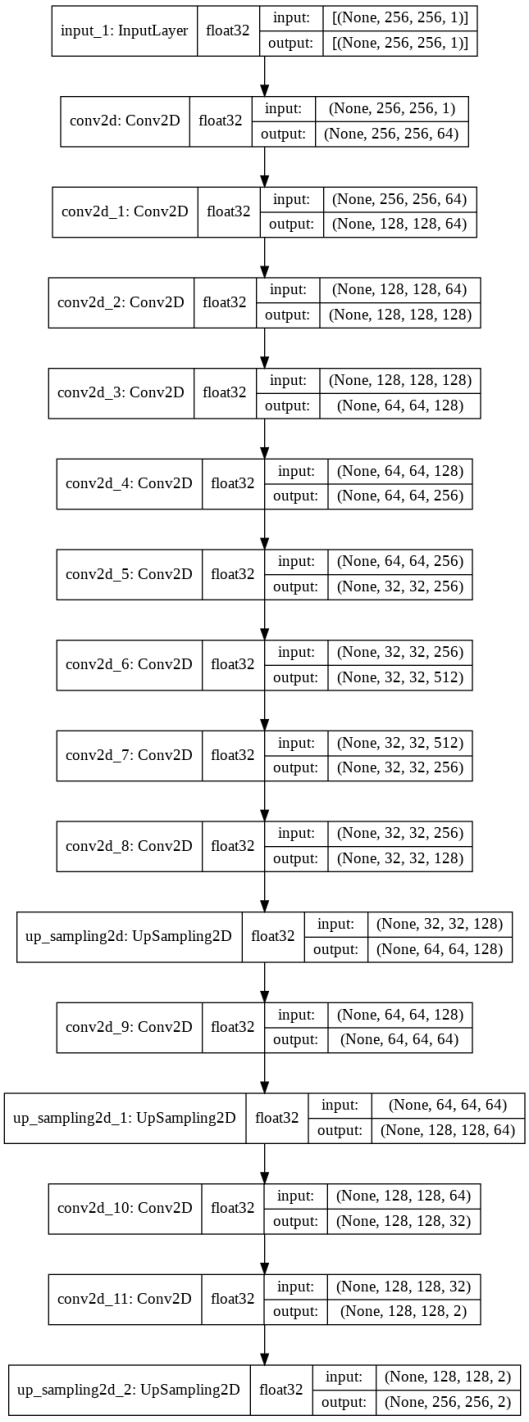
4. Experiments

We want to test the results from that we produce to the CNN proposed in the first paper [9]. This is because the other two papers and most of the image colorization papers compare their results to the results in the paper *Colorful Image Colorization*. The dataset we will use to test the results will the be Flickr Image dataset[5] which can be found in *Kaggle* . This is because the images are in the same category as the images that were used in the first paper. The dataset contains around $30,000$ 3 channel RGB images of pictures posted in *Flickr* which is around $4GB$ in size. The sizes of the images are of different dimensions so either resizing the images or the model to take in any size input will be necessary. We want to evaluate the results by comparing the % difference between the resulting image and the ground truth. Additionally, we want to compare the images we create from the images that have been colorized by the CNN in the first paper[9]. Continuing, the main dataset used in the papers *ImageNet*[2] is only provided to researchers who have directly contacted requesting it and though a small dataset can be found in at *Kaggle* it won't be directly the same as the full dataset. Thus, our choosing of the current dataset instead is due to it striking resemblance towards the *ImageNet*[2] dataset.

4.1. Model Specifics

The initial model is a CNN which takes in a $256x256$ black and white image and attempts to colorize it using the LAB color space addition at the end of the convolution. Though the model is built with the reference of the first paper modifications have been made to attempt to improve the results. There are 9 Conv2D layers in the model in which each alternating Conv2D layer strides of 2 are taken as opposed to the prior layers which perform the convolution with no strides. The opposed model contains 8 layers in total with striding and maintaining the same image ratio from layer 5-8 so just from convolutional layers, our model is already deeper. Additionally, at layer 10 we upsample the image inputted by using a UpSampling2D layer. Activation functions for all layers except the last Conv2D are of that $Relu$ with $same$ padding. Our CNN architecture contains a total of 16 layers as opposed to the 9 layers introduced in the paper[9]. The model was trained with 1000 epochs, 10 steps per epoch, 2 validation steps, and a batch size of 50. Though the model would ideally run for longer issues with higher epochs sizes arouse throughout training issues arouse.

Figure 7. CNN model to colorize images



4.2. Training issues

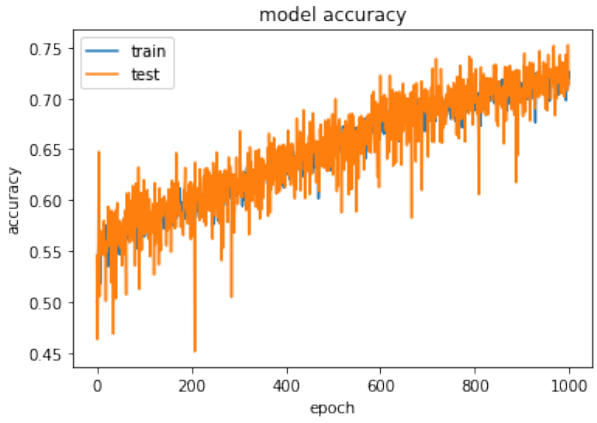
As mentioned ideally the training of the model will be over longer epochs as opposed to the minimal 1000 epochs of this current experimentation. Issues arose with the technology we chose to implement our model and training. Since we used *GoogleColab* we did not have access to

premium GPUs to run our codebase and thus our training consistently failed due to running out of either memory or RAM. Attempting to train the model with $10,000$, $5,000$, and $2,000$ epochs consistently caused failure in the middle of training at random epochs each attempt of training. An attempt to run the codebase on *Jupyter Notebook* with our hardware but collided with issues such as setting up *NVIDIA CUDA* toolkit and its libraries to run our hardware and make it accessible and visible. Attempts to run it on our CPUs were also an option but it took a toll on the training time as it caused the model to train extremely slow when compared to running the book in *GoogleColab*. Taking these variables into consideration we settled to run our model for $1,000$ epochs which were able to successfully train the model in a matter of 4 hours. We believe the root cause of these long training times is the dimension of the images and also equating the LAB color space images.

4.3. Model Accuracy

The model demonstrated a linear trend with accuracy in both the training and testing it. This trend shows promising results moving forward with this model and future work as the law of diminishing returns does not appear to apply as the accuracy tends to increase with the number of epochs the model is trained. From these results, we conclude that the model will ideally want to run for longer epochs until the law of diminishing returns starts to apply as we can determine the limitations of the model we established.

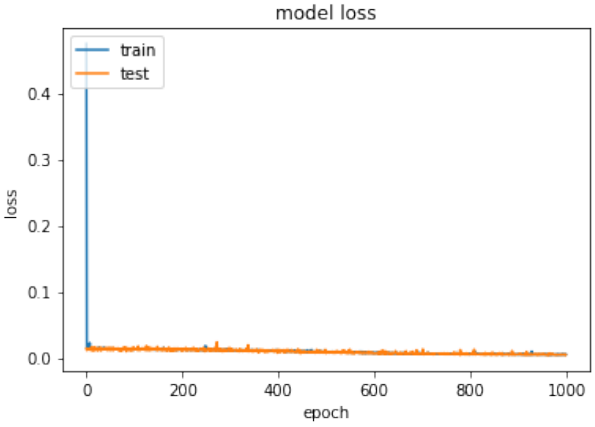
Figure 8. CNN model accuracy



4.4. Model Loss

At the start of training, we notice a high model loss during training with a ceiling between < 0.45 and > 0.5. This could be due to the model initially not knowing much information and the weights being untouched. Towards the 20th epoch, the loss for training goes down dramatically settling at about 0.02 and slowly goes down though it's not very noticeable or significant past this point. The loss during the testing appears to follow this trend from the very start as it starts with an initial loss of about 0.02 as mentioned and slowly decreases as the number of epochs increases.

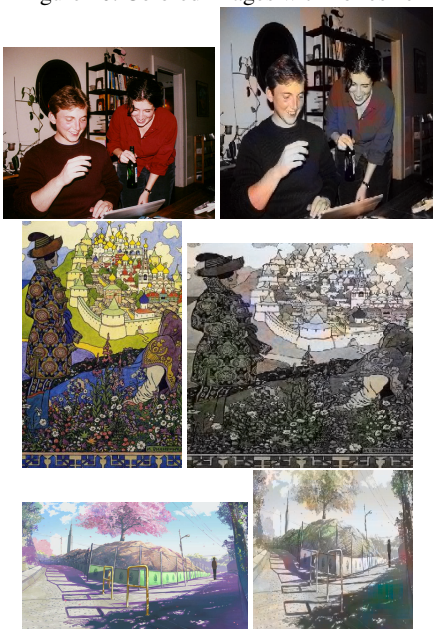
Figure 9. CNN model loss



4.5. Results

We didn't notice any significant differences in the coloration in the three datasets we proposed. We believed that the coloration would differ as the types of the images are of different categories and styles such example is the anime scene when compared to the real-life images the model has been tested on. Ideally, we would've liked to test all datasets and see how each would result as we assumed that it would affect the model's weights and predictions.

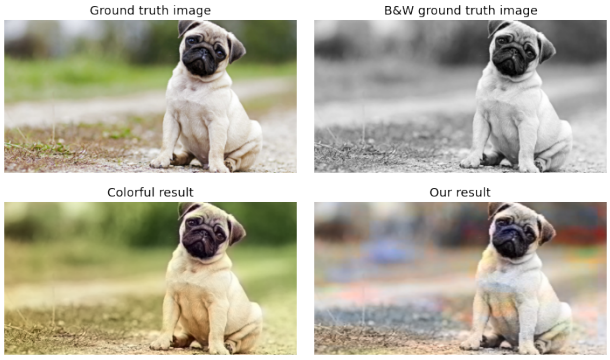
Figure 10. Colored images with no resize



4.6. Comparing results

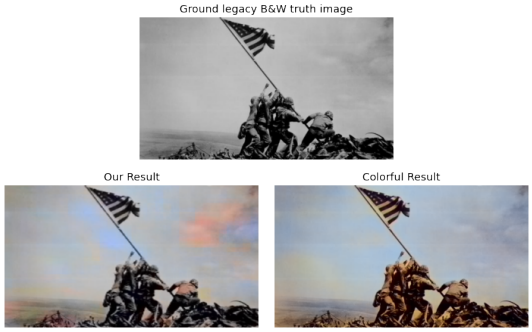
To see how our model will hold up we tested our model to compete with the model proposed in the first paper to see if our model proposed with a deeper network will improve on the original model[9]. Though our model resembled the majority of the colors by qualitative measurements match with slight differences. With we consider this a success by those means but not an improvement on the model we wanted to improve since the colors appear to be more accurate and represent images with more accuracy. We found that the model in the first paper, referenced in the results as *Colorful* to fit the paper title[9], resulted in extremely better results. The table shows the accuracy when the model is compared to ground truth images. The ground truth images have been converted to black and white format from their original RGB format and we determined the percentage difference between the original ground truth image and the colorized image by the models. The following differences were calculates using a random image in which the right end could be the maximum %. Our model is underwhelmed and outclassed by the *Colorful* model but it's good to see some progress was done.

Figure 11. Colored images comparison with resize



Model comparison	
Model	Difference from Ground Truth
Colorful[9]	$94\% + .05\%$
Ours	$96\% + 4\%$

Figure 12. Attempt to color legacy B&W images



Looking at the model comparison table we compared our proposed model with the *Colorful*[9] model. We noticed a difference of on average of **94%** difference from the ground truth image in pixels with cases being of **94.05%** on the highest end. Our proposed model had a difference in pixels of an average of **96%** which is already higher than the highest possible % from the other model. Furthermore, the highest difference that was achieved when testing it multiple times was **100%** meaning that there was not one pixel that was identical on the highest % of the difference. Thus, quantitatively our model performed worse than the established *Colorful*[9] model. Additionally, as we demonstrated the generated colored images in our model are not of the same caliber as the images generated by the other model and even when compared to the ground truth images our model does a poor job to colorize the images though there is some color there. To be able to quantify the ability to tell if the images appear real or fake we would have to survey folks to determine which images they believe are generated and which are *real* though we believe the images generated by our model will not be favored.

5. Limitations and Future Work

Though our model was able to find some success in colorizing images it did not hold up to the same caliber as the *Colorful* model and future improvements could be made to the model to improve the results. For starters, we wish to train our model for longer epochs as we believe that is the root cause of our model not coming close to the same caliber as the *Colorful* model. Additionally, though the model comparison differs from the original with about 94% with the model from the first paper qualitatively the image is on par with the ground truth image thus, the quantitative results are not that impactful. Comparing our model qualitatively speaking it's not close to the ground truth though we are seeing some color being produced in the image.

5.1. Improvements

We hope to use better hardware in the future as opposed to the default GPUs provided in *Google Colab* since we wish to train our model for longer to determine if our model truly is that not on par with the original architecture. Additionally, we wish to train our model using different datasets to determine if different styles of images would affect improvements/regressions in the model accuracy and qualitative elements. As mentioned in the paper to compare the qualitative results we would need to survey people to determine if they believe generated images are indeed real or not using the metric **AMT**[9]. Additionally, we realized late that we should've saved the model to get around the issues that we had with training. Saving the model and it's weights would've allowed us to be able to run the training loop multiple times and load the weights and save the new ones. We

found out about this late but were able to save the weights to be able to load them later and test the model on other images. Moving forward, we could use this method to get around *Google Colab* and it's issues or we could set up our own GPUs in our personal machines to run the training loop for higher.

6. Conclusion

Our deepening of the model did not succeed in being able to improve on the original architecture that was established in the paper *Colorful Image Colorization*. We don't have concrete evidence as the model was run for 1,000 epochs due to hardware limitations. We noticed a linear trend with our model and want to explore better hardware to determine if our model could come to be of the same caliber or be better than the original model. As of now the model was successful in being able to add some color to the images but not to the same degree as being able to colorize the image fully with accuracy qualitative color representation as to the original model.

References

[1] Home - deoldify. https://deoldify.ai/. (Accessed on 04/26/2021).

[2] Imagenet. http://www.image-net.org/index. (Accessed on 04/26/2021).

[3] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao. Deep koalarization: Image colorization using cnns and inception-resnet-v2, 2017.

[4] Danil. Art images: Drawing/painting/sculptures/engravings, May 2018.

[5] Hsankesara. Flickr image dataset, Jun 2018.

[6] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[7] Rhapsody. Japanese anime scenes, Mar 2020.

[8] P. Vitoria, L. Raad, and C. Ballester. Chromagan: Adversarial picture colorization with semantic class distribution, 2020.

[9] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization, 2016.