
# COMP30170 Final Year Project

Deep Colorization

**Qinyuan Zhang** (UCD: 15205944)

Report 3, April 28, 2019

---

**Table of Contents**

# 1 Abstract

Colorizing grayscale images is a subject in the area of image processing, while machine learning with the multi-layer neural network is popular nowadays. After some reviews of previous and recent researches in the area of image colorization, three different experiments were carried out in this paper while all of which employed deep learning. Architectures, training strategies, and results of all three will be demonstrated. Comparing to some previous methods which do not apply deep learning, these end-to-end approaches are totally automatic. Besides, they are also faster and more efficient. A comparison section and a user study are also included to assess how accurate these results are and whether these outputs could actually confuse man-kinds.

# 2 Introduction

Deep-learning is one of the most popular research areas among the machine learning subject. Large datasets for supervised training of deep neural networks along with powerful GPUs are essential for such achievements. By mapping and calculating data on multi-layer neural networks, researchers are able to solve different kinds of complex problems which are hard or even impossible to solve in the past. On the other hand, Image processing refers to the area where computer algorithms are used to analyze and process digital images. There are many promising outputs created by applying deep-learning techniques on image processing these days. Tasks like image identification, classification or style transferring are providing great evidence that this could bring better results comparing to some older approaches.

This project aims to explore the deep-learning techniques on the subject of image colorization where researchers are devoted to colorize grayscale or black & white images using existing datasets or pictures online. Artist before use their own experience to colorize images by applying colors on each and every pixels. With the developing of computer techniques, researchers like Levin, Huang and Yatziv [1–3] introduced the scribble-based approaches where few scribbles are enough to colorize a whole image. Example-based colorization is also raised by Welsh, Gupta and Chia [4–6] where the similar image needs to be found before the colorizing process and colors will be transferred to target image. Despite the fact that these approaches could provide great outputs, they all depend on human intervention or similar images. Therefore, these approaches are considered slow to process and highly restricted by external factors.

A fully-auto "learning-based" approaches are crucially useful to solve the problems raised by the others before. In order to create these sort of algorithms, the trending convolutional neural network is employed. Cheng and Schmidhuber [7,8] firstly address the possibility to efficiently create colorization outcomes by using convolutional neural networks. However, simple usage of this architecture might lead to a brown-ish hue in the results across Lab color space, which is also known as "Sepia Toning". In order to provide better performance, several other techniques are combined with the convolutional neural network. These include color prediction layers [9], real-time human guided [10], transfer-learning [11, 12] and etc.

Large datasets are critical in the deep-learning procedure. Over one hundred thousands images are prepared from Cifar-10 [13] and ImageNet [14]. Experiments are based on three different approaches, which is one single convolutional neural network, a combination of single convolutional neural network and ResNet classification layers [11], and generative adversarial network (GAN) [15]. Some of the results are demonstrated in Figure 1. Additionally, to prove that deep-colorization is not a

hardware intensive and time-consuming process, simple architecture training on a small dataset is also implemented and some of the results are shown as well.

The major contributions of this paper are as follows:

- Research on previous and recent approaches related to image colorization area.
- Explore and prove the feasibility to colorize grayscale images by applying three different deep-learning approaches.
- Analyze the convincing rates of the colorized images based on a survey among dozens of man-kinds.

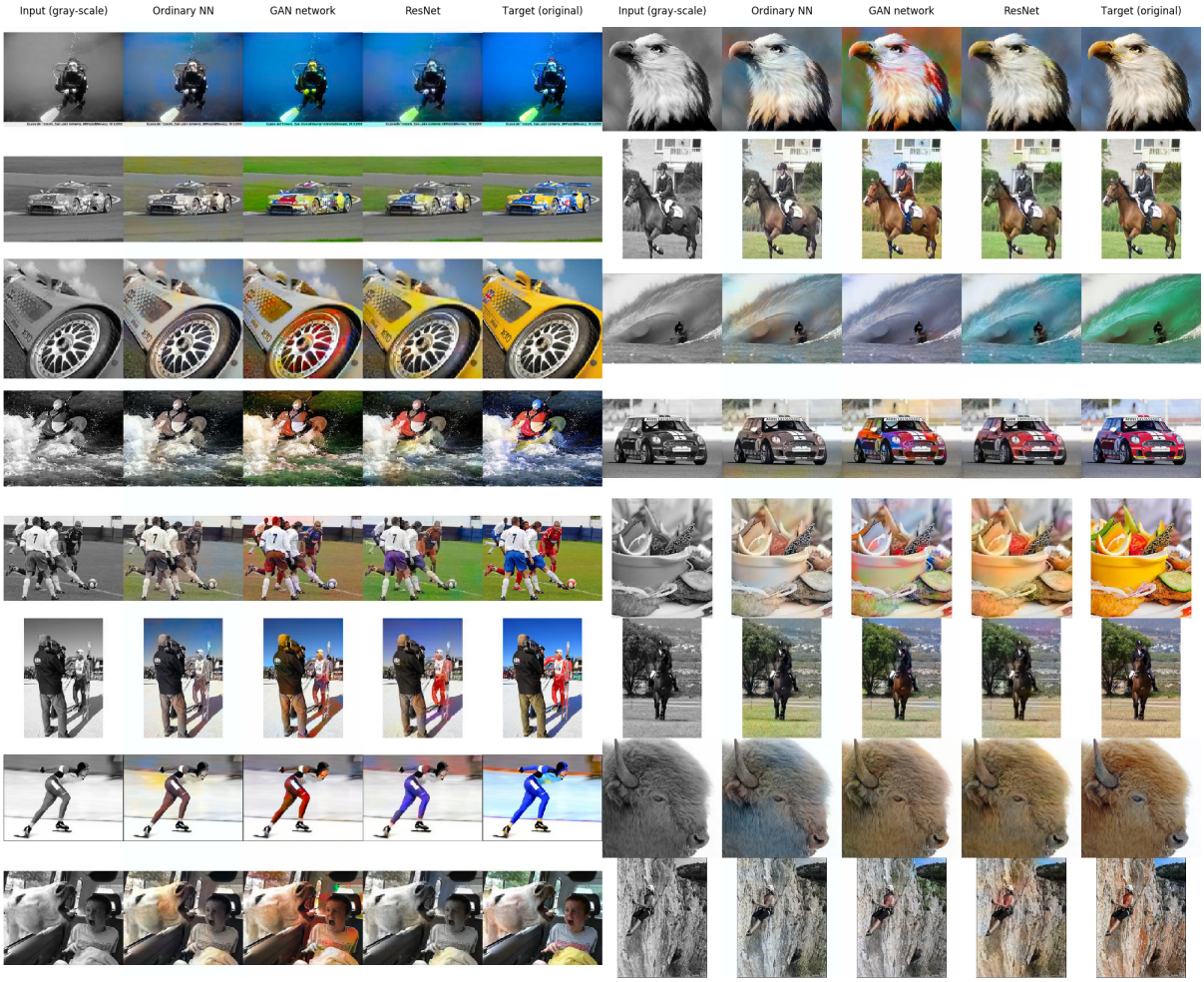


Figure 1: Some of the colorization results from the experiments. Grayscale input images are listed on the first and sixth columns. Three different approaches' outcomes are listed on columns second to fourth, seventh to ninth. At last, the real colorful images are listed on columns fifth and tenth.

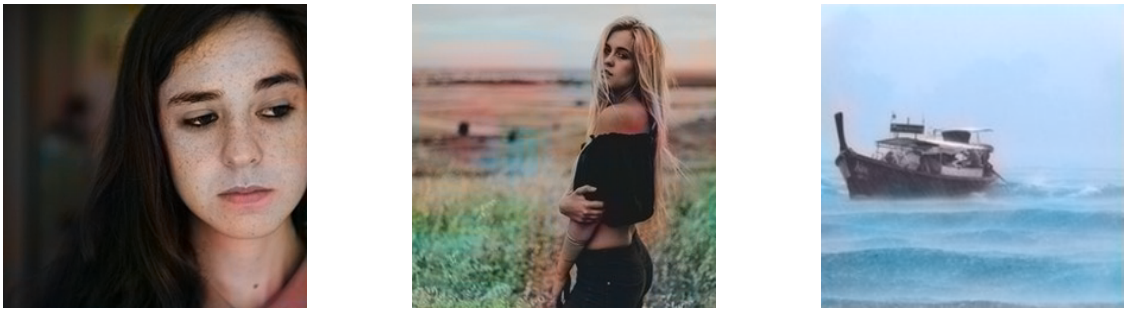

Figure 2: Some of the outcomes created by using small database and less epochs.

# 3 Related Work

There are three main kinds of colorization approaches currently in this research area. A brief overview of all three previous studies will be listed and illustrated in this section.

## 3.1 Scribble-based Colorization

Levin et al. [1] firstly introduced a way to colorize images without putting works on each and every pixels. Human-annotated color scribbles together with optimizer algorithm can colorize nearby pixels with similar input intensities. Yatiz et al. [3] raised a similar method where human scribbles are considered in different weights in the process of generating results. Huang et al. [2] created an algorithm which can auto-detect the edges on the images, therefore, create better colorization on the objects' boundaries. Qu et al. [16] as well as Luan et al. [17] use semantic features to reduce the needs of user scribbles.

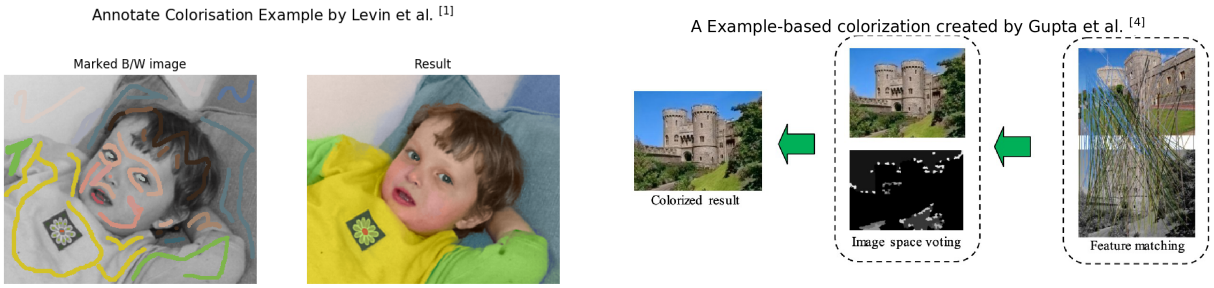


Figure 3: Demonstration of two approaches regards to Scribble-based and Example-based Colorization.

## 3.2 Example-based Colorization

These example-based colorization approaches do not need human intervention. Instead, they use similar images' colors information to colorize target one. In general, colors are transferred from one to another. Two main categories can be identified based on the different sources of the reference pictures.

### 3.2.1 Human-feeding Examples

Welsh et al. [5] firstly introduced a colorization method based on pixels analysis where different aspects of parameters like intensities, nearby pixels' similarities are taken into consideration. In the result, transfer the colors for every pixel on the target image from the most similar pixels on the reference image. Charpiat et al. [18] created a global optimizing method to transfer colors automatically. Similar methods can also be found from Gupta [4]. These approaches highly limited by the quality and similarity between the example images and target images. Meanwhile, finding promising example images are significantly time-consuming and unpredictable.

### 3.2.2 Internet-based Examples

In order to improve the efficiency and accuracy of selecting reference images, Chia et al. [6] proposed a solution which takes the semantic text labels of the foreground objects on the scene, the system will find similar images online, download them, then, transfer the colors. Similarly, Liu et al. [19] use many online images for one object to colorize. The algorithm can "recover an illumination-independent intrinsic reflectance image" for the target scene from multiple references. However, this is restricted to certain pictures with certain foreground objects and clear shape (e.g famous landmarks).

## 3.3 Learning-based Colorization

Both scribble-based and example-based methods have their limitations. They are extremely time-consuming and highly restricted by external factors like scribble's accuracy and reference images' qualities. Due to these facts, learning-based colorization with the convolutional neural network (CNN) becomes popular among relative researchers. Cheng et al. [7] firstly addressed that the CNN can provide great outcomes and it is efficient on various scenes. However, this or similar approaches usually create brown-ish outputs due to the fact that brown is the most similar colors to any other natural colors, therefore, creates the smallest average errors in the CNN.

In order to create a better fitting, Zhang et al. [9] employed a separate color prediction layer just before the output layer. To simplify the process, only 313 colors are selected and each pixel needs to fit in one greatest possible color. This approach gives better outputs due to the fact that it generates better diversity among the color space. In addition, they also introduced a real-time user guide colorization approach [10]. By using deep learning and humans intervention, the colorizing procedure can become faster and more accurate.

Some other fully-auto deep learning based methods are also introduced. Baldassarre et al. [11] and Iizuka et al. [12] use the pre-trained weights to get the classification information of the grayscale input, then, concrete with the output of the CNN's encoder before pushing to the decoder. These transfer learning approaches take the advantages of the high-level features extracted by previous works and get benefits on low-level vision domains (similar to some image enhancement and edge detection approaches).

Generative adversarial network (GAN) [15] is a trending new architecture where it shows great powers to generate new contents. Two separate trainable networks working together can provide promising results in many vision areas like objects transferring, style transferring, and etc. Recent researchers like Nazeri et al. [20] are trying to use GAN on image colorization as well.

# 4 Method

A deep-learning neural network uses the data from the training datasets as the input/output. It calculates and minimizes the loss between the predicted output and the ground truth again and again to achieve the goal of being more and more accurate. This project will use the convolutional neural network which typically extracted the high-level features from the original datasets and then decodes to a low-level specific form of outputs. This paper will discuss the input/output and layers selection for this project first. Then it will move on to the network's choices for loss, optimizer and other parameters.

## 4.1 Input and Output

Firstly the input and output shapes or formats need to be specific for this project. If we want to colorize a grayscale or luminance pictures, we need to input their grayscale pixels' values and output the chrominance values of all the pixels. Additionally, no color information should be passed as inputs. In this case, we better perform our pictures in the Lab color space instead of the RGB since the former color space separate the information to lightness component (which stands for the "L" in equation (1)) and color ones (represented by greenred and blue-yellow as C in equation (2)).

$$L \in \mathbb{R}^{H \times W \times 1} \tag{1}$$

$$C \in \mathbb{R}^{H \times W \times 2} \tag{2}$$

Many relative approaches have a similar choice as this decision [7, 9–12]. However, different color spaces like the YUV are also useful. For example, the YUV color space was employed in Cheng's paper [7]. In the YUV color space, "U" and "V" represent the chrominance information while the "Y" is the luma component (the brightness).

## 4.2 Convolutional Neural Network

Once we figure out our input and output, we need to look at the layers inside our convolutional neural network. Basically, we want to find a map $f()$ from lightness input L to chrominance output C, which is shown in equation (3).

$$\widehat{C} = f(L) \tag{3}$$

There are two main parts inside our neural network, which could be understood as encoder and decoder [11, 12]. Dim reduce is necessary because features are able to be extracted and calculation size will be reduced which can prevent overfitting. The encoder uses strides or pools to extract the features form our images while the decoder uses up-sampling to generate suitable outputs.

Convolutional 2D layers are added in both two parts of the layers with the ReLU activation function. Other functions like Sigmoid transfer function are also useful in some papers like Iizuka's one [12]. It should be noticed that one of the advantages of ReLU activation function is the reduced likelihood of the gradient to vanish. On the other hand, Sigmoid() activation function creates a lot of non-zero results in dense representations, which is a disadvantage. Based on the two reasons above, ReLU is mostly used in the following attempts. Notice that a lot of the previous approaches do not use max-pooling layers due to the fact that it reduces the data size too aggressively. It will lead to some performance issues which were mentioned in both Iizuka and Zhangs' papers. [9, 12].

## 4.3 Loss

The most common loss measuring method is the "Euclidean loss between predicted and ground truth colors" [9]. A similar method like "least squares minimization" is also useful [7]. In practice, we use the "mean square error"(MSE) loss measurement in the Keras or TensorFlow, which is shown as formula (4),

$$\text{MSE}(X) = \frac{1}{2HW} \sum_{k \in \{a,b\}} \sum_{i=1}^{H} \sum_{j=1}^{W} (C_{k_{i,j}} - \widehat{C}_{k_{i,j}})^2 \tag{4}$$

where $X$ represent the image itself, $H$ and $W$ stand for the height and width of the image, $C_{k_{i,j}}$ and $\widehat{C}_{k_{i,j}}$ denote the ground truth and prediction color value on each pixel and channel. The reason why this is employed is that it provides a straight forward way to measure the outputs' performance. Previous works also suggested that it can provide promising results. "Mean absolute error" in equation (5) is very similarly to the previous loss function which was listed before. It calculates the mean absolute errors between the predicted results and the ground truth.

$$\text{MAE}(X) = \frac{1}{2HW} \sum_{k \in \{a,b\}} \sum_{i=1}^{H} \sum_{j=1}^{W} |C_{k_{i,j}} - \widehat{C}_{k_{i,j}}| \tag{5}$$

## 4.4 Optimizer

In the experiments, Adam [21] is mostly employed. This approach is widely used in many image colorization tasks [9, 11, 12, 20]. However, the initial rate might vary, for the first two approaches, $n = 0.001$ is chosen while in the last approach, smaller initial rates will provide better outcomes. In the GAN approach raised by Nazeri [20], it also introduced Sigmoid Cross Entropy with Logits as the optimizer for the discriminator.

## 4.5 Other Techniques

In order to improve the performance of the results and bring better outcomes, several additional techniques are employed in personal experiments.

### 4.5.1 ResNet Inception

Pre-trained models such as Inception [22] and ResNet [23] can always provide high-level features extracted from the target images, which can benefit our subject. Addressed by Baldassarre et al. [11] and Iizuka et al. [12], with the additional classification information, the colorization process could greatly benefits. Inception-ResNet-v2 is introduced in the experiment.

### 4.5.2 Generative Adversarial Network

Goodfellow et al. [15] proposed the Generative Adversarial Network where two small networks called "generator" and "discriminator" could be trained together to generate persuasive outputs. The training process will hold one network first, train the other network. Then, the position will be switched. The generator's responsibility is to create fake outputs while the discriminator should distinguish real or fake results. By training both networks, in the case, keeping the game going, the generator will provide convincing outputs. Mathematically, the overall cost function of the process can be expressed as

$$\min_G \max_D V(G,D) = \mathbb{E}_x[log(D(x))] + \mathbb{E}_z[log(1 - D(G(z)))] \tag{6}$$

The cost function of the discriminator, in this original approach, can be represented as

$$\max_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \max_{\theta_G} \mathbb{E}_z[log(1 - D(G(z)))] \tag{7}$$

However, due to the fact that the convergence rate will "tremendously slow down" [20] while generator continuously produce similar results if the discriminator performs well, a redefined discriminator's cost function is raised and could be demonstrated as

$$\max_{\theta_G} J^{(G)}(\theta_D, \theta_G) = \max_{\theta_G} \mathbb{E}_z[log(D(G(z)))]. \tag{8}$$

# 5 Experiments

There are three main attempts regarding the deep-learning colorization carried out in this section. All of them are based on Keras or TensorFlow.

## 5.1 Database

The database is collected from ImageNet [14] Fall 2011 dataset, where 50,000 images are gathered, 6.0 GB in total.

The testing database is from ImageNet Fall 2011 dataset as well. 975 images are gathered in total.

## 5.2 First Attempt

The first attempt used a single convolutional neural network where the whole colorization process is in an end-to-end form.

### 5.2.1 Architecture

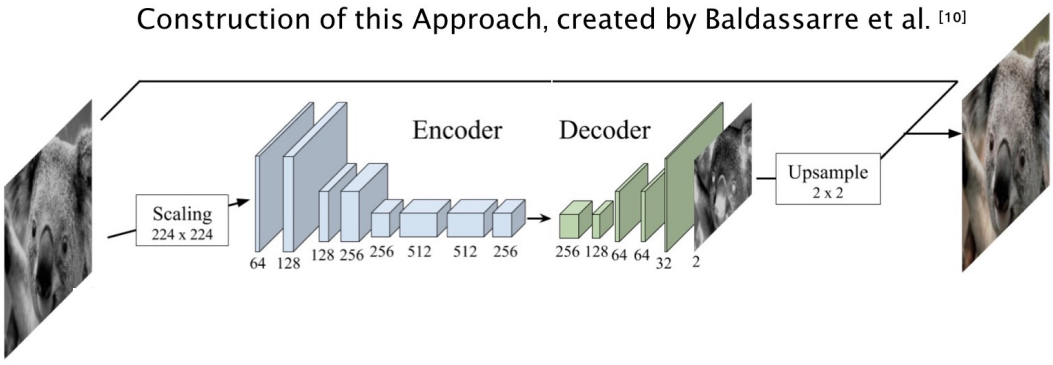


Figure 4: The network's construction of this attempt.

The CNN from Baldassarre et al. [11] approach is employed in this experiment. Logically this network could be divided into two parts, which is Encoder and Decoder. The encoder extracts features from the input while the decoder will try to map high-level features back to low-level vision. Input and output's form is discussed in section 3.1 and 3.2. $256 \times 256$ images from bigger training dataset are treated as inputs firstly.

| Encoder Layer (type) | Output Shape | Decoder Layer (type) | Output Shape |
|---|---|---|---|
| input 1 (InputLayer) | (None, 256, 256, 1) | conv2d 8 (Conv2D) | (None, 32, 32, 128) |
| conv2d 1 (Conv2D) | (None, 128, 128, 64) | up sampling | (None, 64, 64, 128) |
| conv2d 2 (Conv2D) | (None, 128, 128, 128) | conv2d 9 (Conv2D) | (None, 64, 64, 64) |
| conv2d 3 (Conv2D) | (None, 64, 64, 128) | up sampling | (None, 128, 128, 64) |
| conv2d 4 (Conv2D) | (None, 64, 64, 256) | conv2d 10 (Conv2D) | (None, 128, 128, 32) |
| conv2d 5 (Conv2D) | (None, 32, 32, 256) | up sampling | (None, 256, 256, 32) |
| conv2d 6 (Conv2D) | (None, 32, 32, 512) | conv2d 11 (Conv2D) | (None, 256, 256, 16) |
| conv2d 7 (Conv2D) | (None, 32, 32, 256) | conv2d 12 (Conv2D) | (None, 256, 256, 2) |

- The encoder takes a bunch of $H \times W \times 1$ images as inputs and outputs features with a size of $H/8 \times W/8 \times 512$. Eight convolutional layers are used with $3 \times 3$ kernels with padding, which can preserve the layers input size. Additionally, a stride of 2 is also applied in the first, third and fifth layers, halving the dimension and therefore, reducing the number of computations required [11].
- The decoder takes the encoder's outputs as inputs. Three up-sampling layers are introduced to counter the three strides in the encoder. After a series of convolutional layers, the decoder will obtain results with color information $H \times W \times 2$.

- Each layer uses a ReLU activation function, exception the last layer which employed the hyperbolic tangent function.

### 5.2.2 Training strategy

MSE, which is discussed in section 3.3, is employed in this approach. Adam [21] with the initial rate at 0.001 is introduced as the optimizer. Training is based on 50,000 images while 50 images are treated as one batch. 25 epochs are done. The early-stop yield the procedure after three epochs when validation loss is not dropping. In total, 25,000 total steps. Took around four and a half hour on a GTX980, i5-6500 with 16G memory.

### 5.2.3 Results

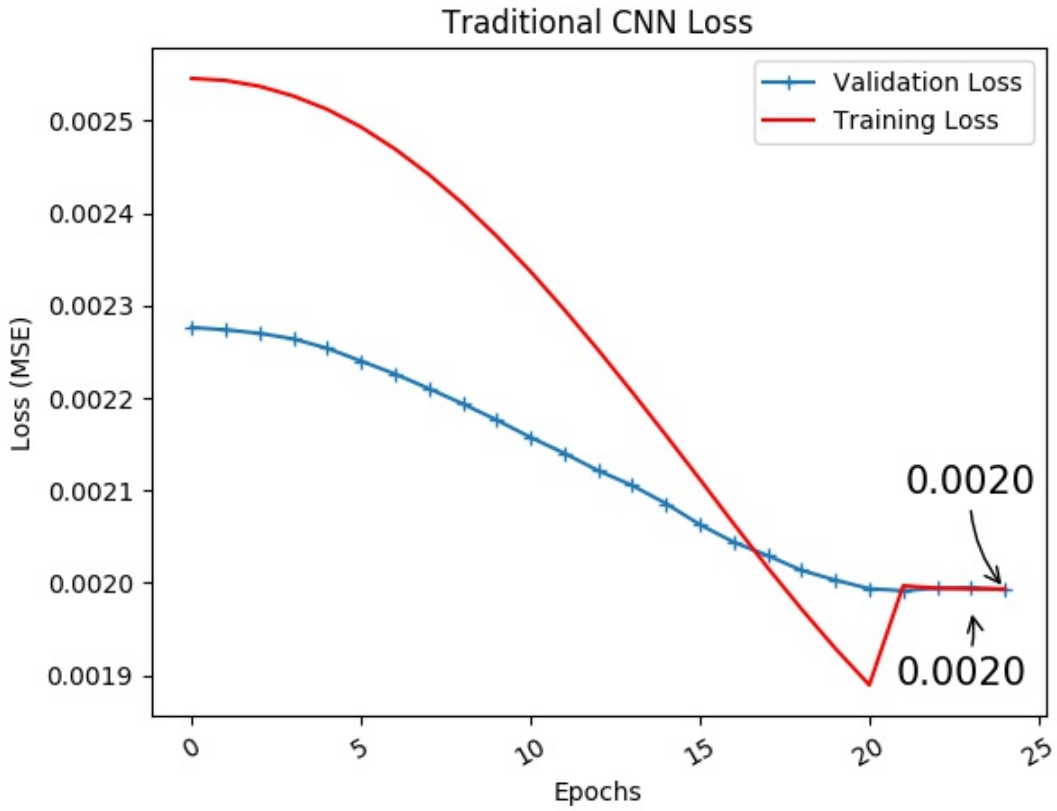


Figure 5: The loss values go down during the procedure. Validation and training loss reach 0.002 at the end.

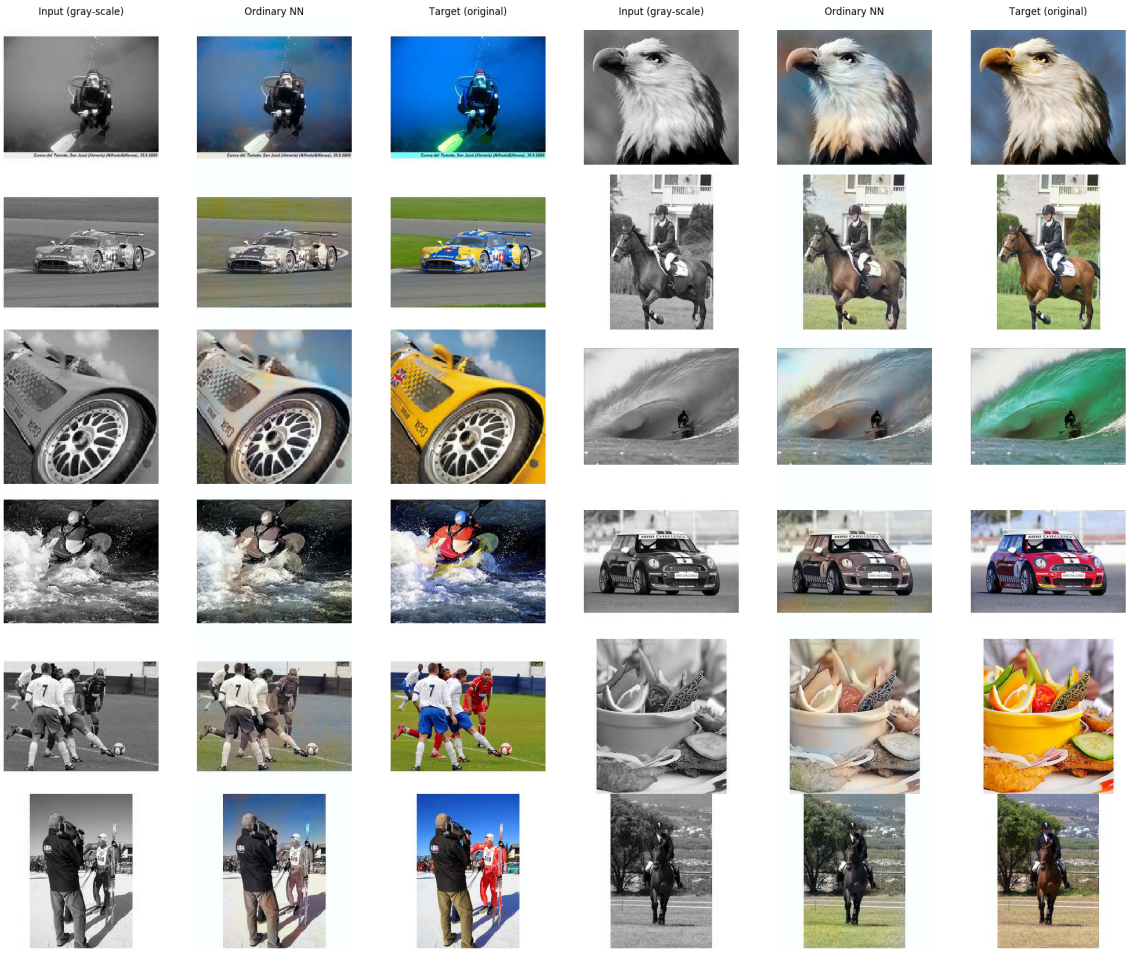


Figure 6: Some of the results. Input grayscale images on the first and fourth column, while ground truth on the third and sixth column. Colorized images are on the column two and five.

Figure 6 shows that in many results, a brown-ish hue might be significant. The reason is that brown is the most similar color to all other colors, therefore, creates the smallest average error value in the MSE.

In order to create better outputs, some additional techniques should be added. And that is what will be demonstrated in the second attempt.

## 5.3 Second Attempt

The second attempt is based on the paper of Iizuka and Baldassarre [11,12]. In their papers, a fusion layer is added between the encoder and decoder. It is inspired by some of the recent researches which are related to transfer learning.

In this approach, Inception-ResNet-v2 neural network is used as the fusion layer which can classify the images. It is concreted into the architecture between the encoder and decoder (which is the gray part in Figure 5).

This approach will create a much colorful result comparing to the first one. That is because more information is taken into the procedure. In this case, it is the classifications of those images.

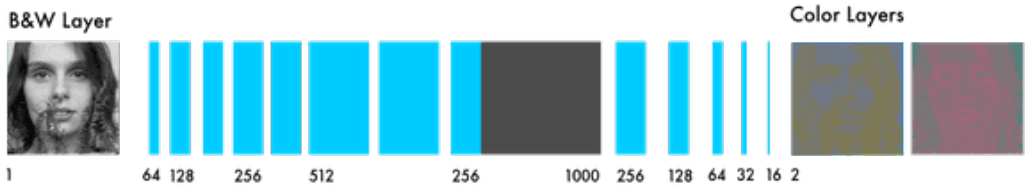


Figure 7: Layers explanation by Wallner [24]

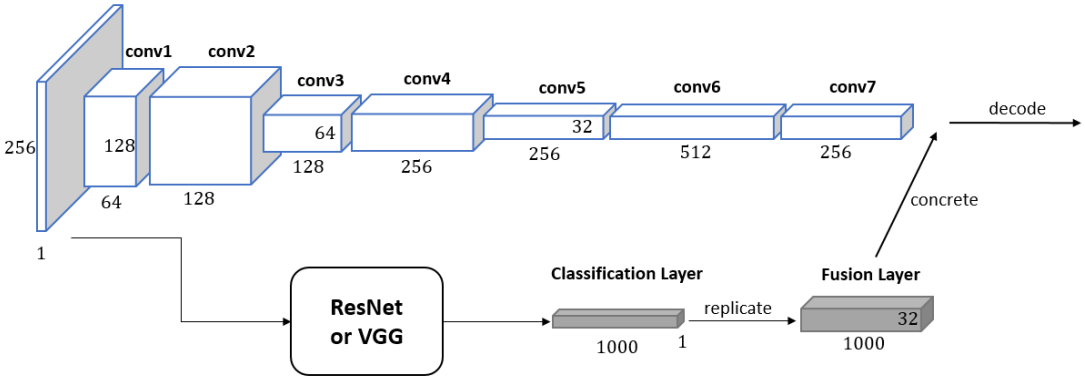


Figure 8: The encoder and fusion's construction of this attempt.

### 5.3.1 Architecture

Shown in Figure 8, the main difference between this and the first approach is the add-on classification and fusion layer.

In order to feed the grayscale input images to Inception-ResNet-v2 model, which is trained by using colorful images with different resolution, some pre-processing should be done. Firstly, images with size $256 \times 256$ will be resized to $299 \times 299$. These $H \times W \times 1$ images need to be put back to $H \times W \times 3$ as grayscale images to fit the pre-trained model's preference.

After the outcomes of the pre-trained model are generated, with the size of $1 \times 1 \times 1000$, it will be replicated to fit the size of the encoder's outputs, therefore, has a size of $H/8 \times W/8 \times 1000$. Lastly the fusion layer will concrete them before put it to the decoder.

### 5.3.2 Training strategy

Firstly, 50,000 colorful images with 256 by 256 pixels are collected as the training dataset. All of these will be resized and put into Inception-ResNet-v2 firstly and stored their outcomes. That is because getting classification outcomes during the training is too resource-consuming. Training epochs are 100 and it took around 18 hours on a powerful platform (Core I5 and GTX1080). Loss function, Optimizer, and other parameters are the same as the first attempt.

### 5.3.3 Results

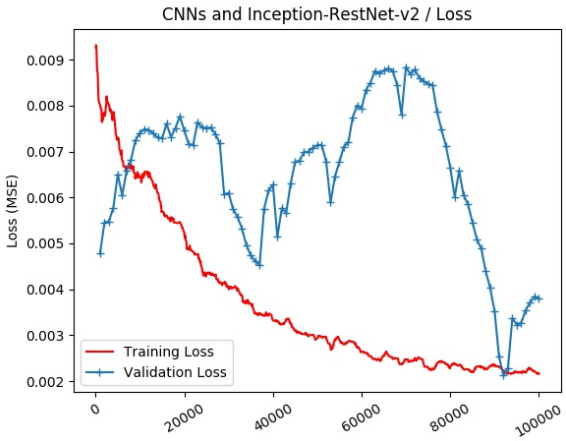


Figure 9: Training loss drops significantly in this approach. Input grayscale images on the first and fourth column, while ground truth on the third and sixth column. Colorized images by applying this approach are on the column two and five.

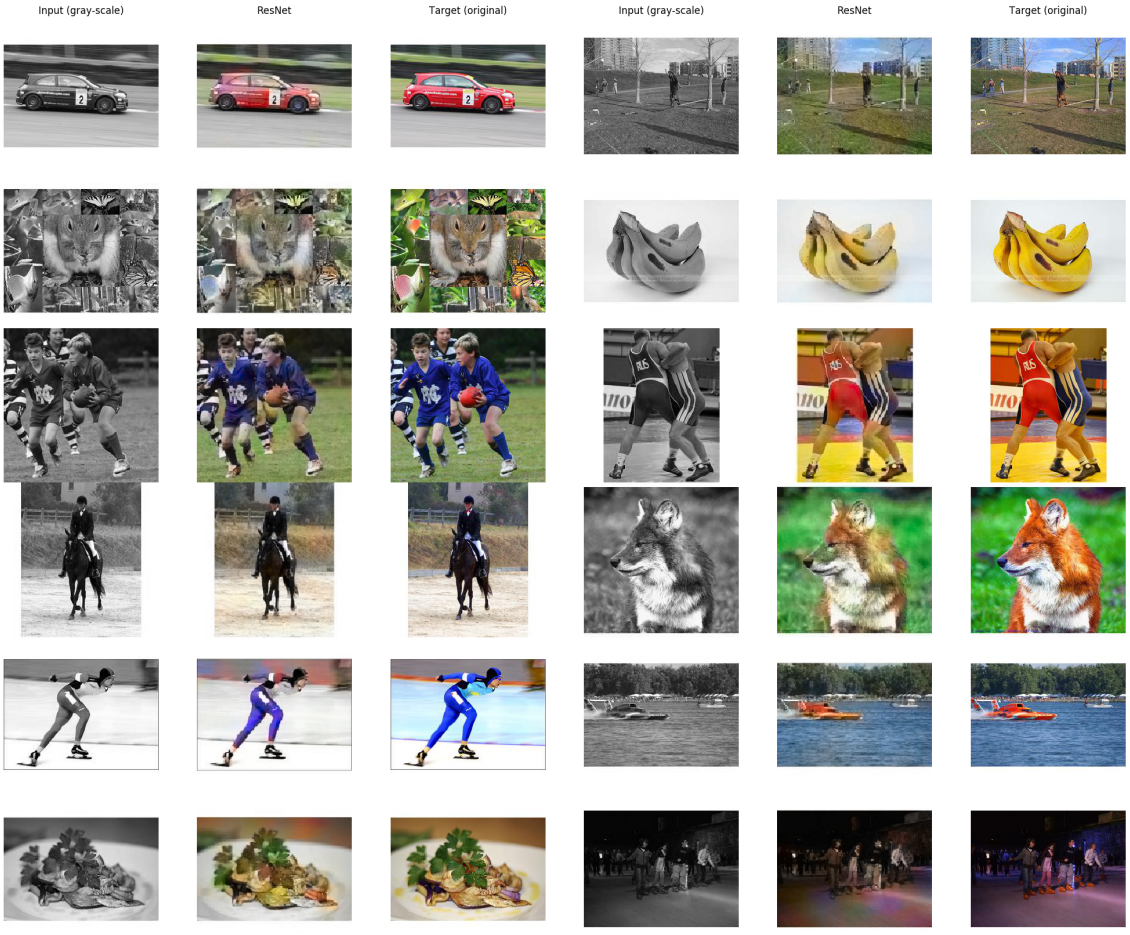


Figure 10: Some of the results. Input grayscale images on the first and fourth column, while ground truth on the third and sixth column. Colorized images are on the column two and five.

It is clear that the results are much more vivid and accurate after 100 epochs comparing to the first attempt's results. The loss reduction can be found quite efficient as well as it is plotted in Figure 9.

Despite the fact that some outcomes seem promising, this attempt still has some flaws. First and foremost, the training procedure is longer comparing to the first one. Next, ResNet is designed to classify colorful images while in our attempt, the target images are not. In that case, these target images may let the classification embed layer creates unreasonable results, which leads to bad performance. At last, the overall architecture and epochs number of the network is not tested as the most efficient or well-performed. Further investigation might need to improve on these three points.

## 5.4 Third Attempt

Lastly, the trending GAN network is introduced for the colorization task. Experiment follows Nazeri et al. [20].

### 5.4.1 Architecture

A typical GAN architecture can be found in Figure 11. In this attempt, generator and discriminator are employed from Deep Convolutional GANs (DCGAN) [25]. Specifically, the generator is a U-Net with an input size of $256 \times 256$.

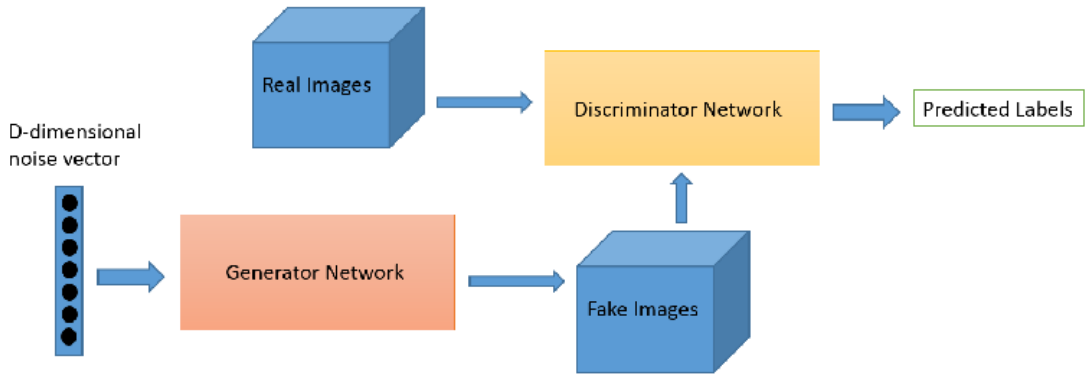


Figure 11: The construction of a typical GAN network. Created by Mirza et al. [26].

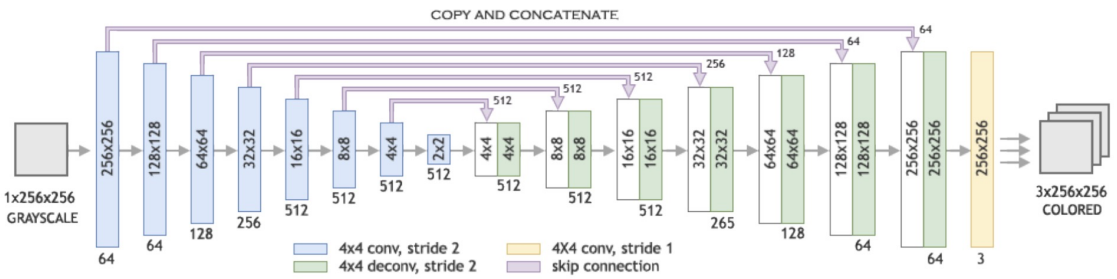


Figure 12: The U-Net is introduced as the generator. Created by Nazeri et al. [20].

For the discriminator, similar architecture from DCGAN is used: a series of 4 4 convolutional layers with stride 2 downsample the input. All layers in the discriminator contain batch normalization, as well as leaky ReLU activation with slope at 0.2. A convolution layer is applied to map to a one-dimensional output after the last layer. A Sigmoid function is followed and its outcome will represent whether the image is real or fake. The input of the discriminator is colorful images either coming from the generator or directly from the training dataset.

### 5.4.2 Training strategy

Similar training dataset is used in this approach. Adam [21] optimizer with a smaller and changing rate is employed. For the generator, MAE is introduced to measure the performance while Sigmoid Cross Entropy is used for the discriminator. Overall time is around 26 hours to train only 20 epochs while due to the complex architecture and limited GPU's memory, only 16 images can be trained as a batch.

### 5.4.3 Results

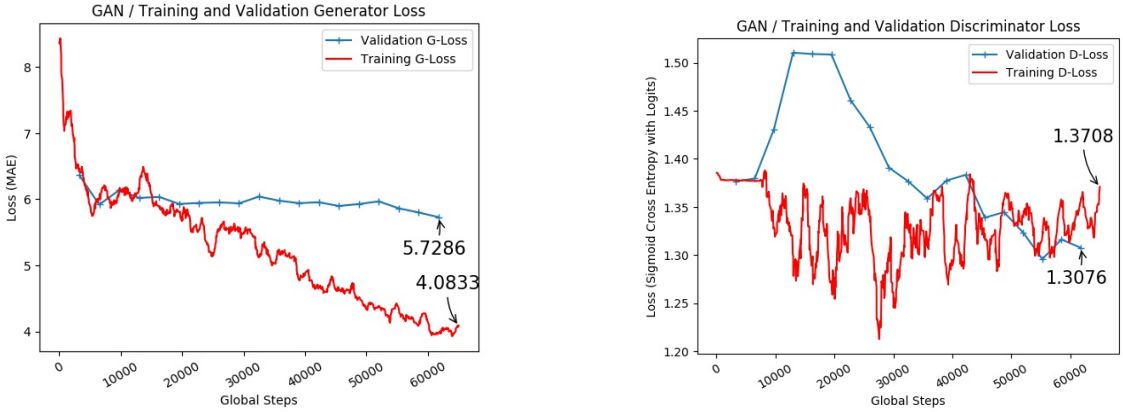


Figure 13: Generator and discriminator's loss values during the training process.

This approach tempts to provide the most saturated results among all three. However, colors are not always accurate, especially comparing to the second approach. Unfortunately, this is also the most time-consuming approach among all three while after a lot of modification and refactoring, it still can not bring competitive outcomes. This probably does not fully implement the GAN's potential.

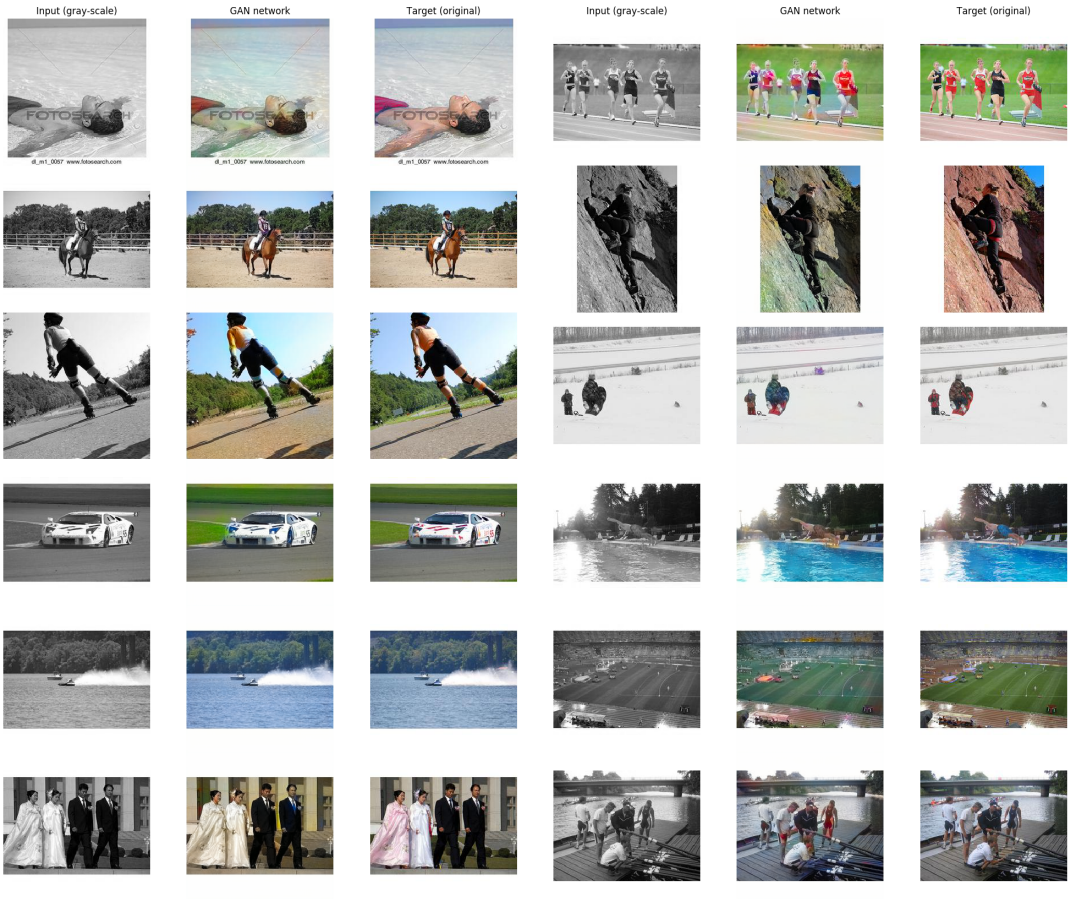


Figure 14: Some of the results. Input grayscale images on the first and fourth column, while ground truth on the third and sixth column. Colorized images are on the column two and five.

## 5.5 Comparison

An accurate formula is employed to measure the color accuracy among all the outputs of three attempts and their ground truths. This equation is represented as

$$\text{acc}(C, \widehat{C}) = \frac{1}{2HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \prod_{k \in \{a,b\}} 1_{[0,\epsilon]}(|C_{k_{i,j}} - \widehat{C}_{k_{i,j}}|) \tag{9}$$

, where $1_{[0,\epsilon]}()$ is an indicator function. Other parameters are the same as they are in section 3.3. By setting $\epsilon$ at $2\%$, all three attempts' color accuracy are demonstrated down below.

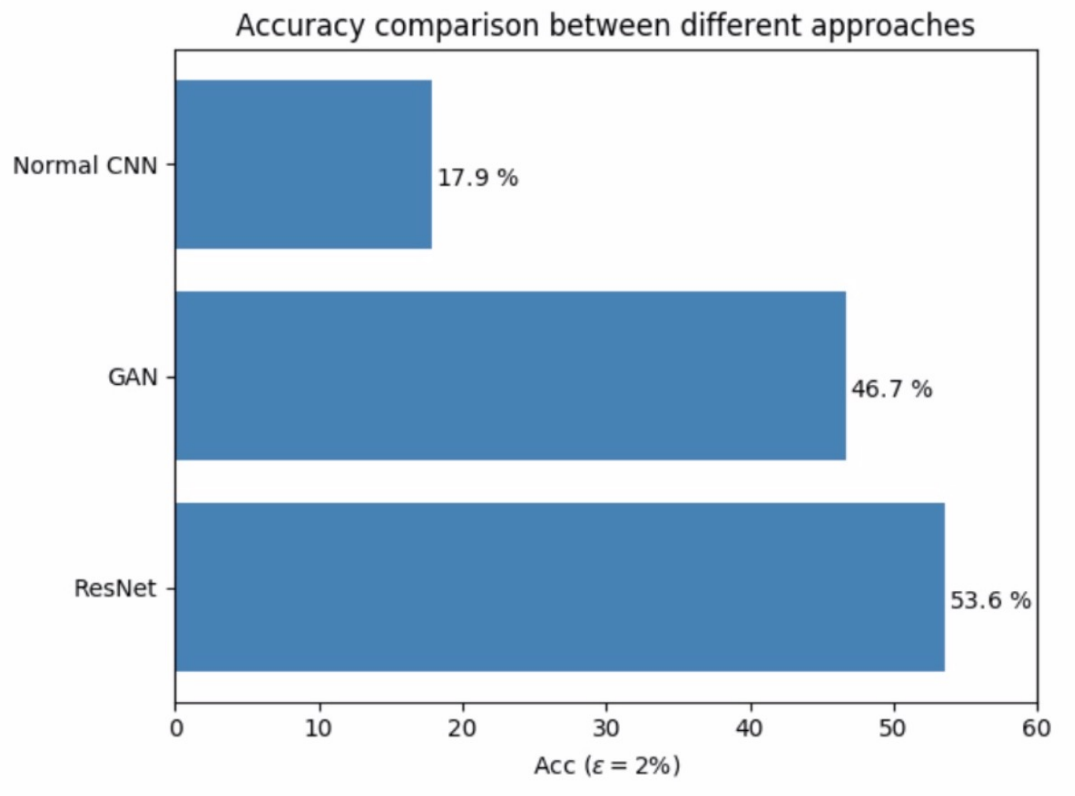


Figure 15: Accuracy among different approaches. Clearly, we can tell that Normal CNN, which is the first attempt, has the lowest accuracy, while the second attempt has the highest.

A more straight forward comparison of three different approaches is shown in Figure 1 where 16 images are selected and three approaches' outcomes and their ground truth are put side by side.

# 6 User Study

To test how the results of three different methods can actually convince human-kinds, a web application is created where survey could be done among students. One hundred randomly selected images from test dataset, as well as their three different outputs by applying different approaches are stored in the website's database ($4 \times 100 = 400$ in total). For part one in the survey, five random pictures are randomly selected and users need to choose whether it is a real image or a colorized image by an algorithm.

Part two allows users to give marks to five randomly selected images. One star refers to no color at all, two stars refer to a few colorization on some parts of the images. Three stars refer to fully colorized but not saturated, while four represent that the colorizing program is perfect.

There are 34 users who took part in this survey.

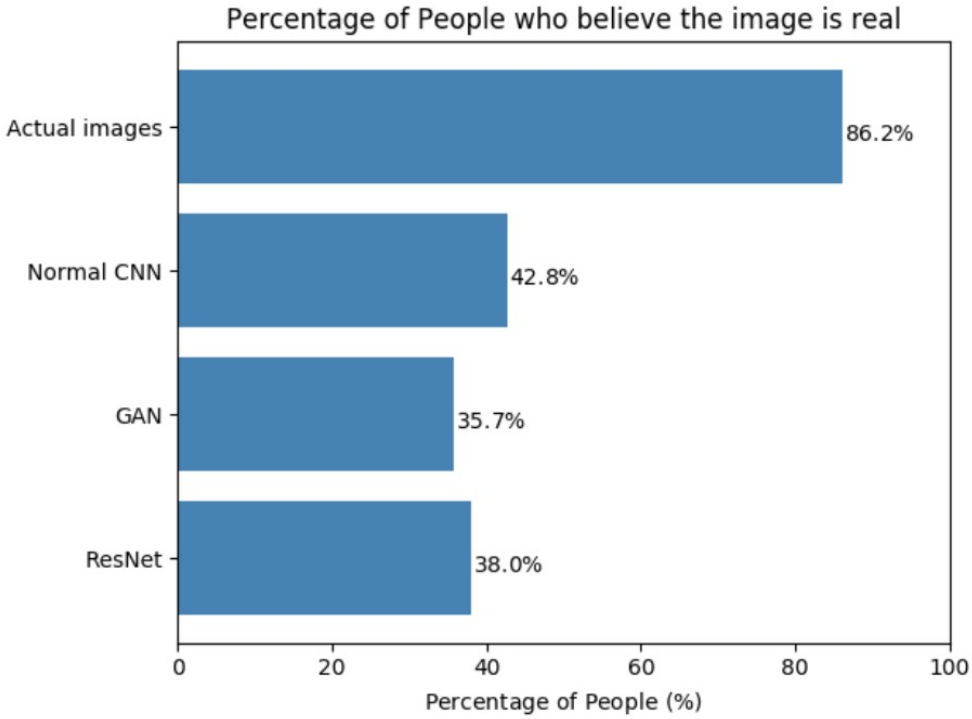


Figure 16: Nearly half of users believe the normal CNN's output images are real, which is a little bit surprising. GAN's outputs are normally too saturated which might give it a drawback. Around 13 percent of people believe a real image is not real.

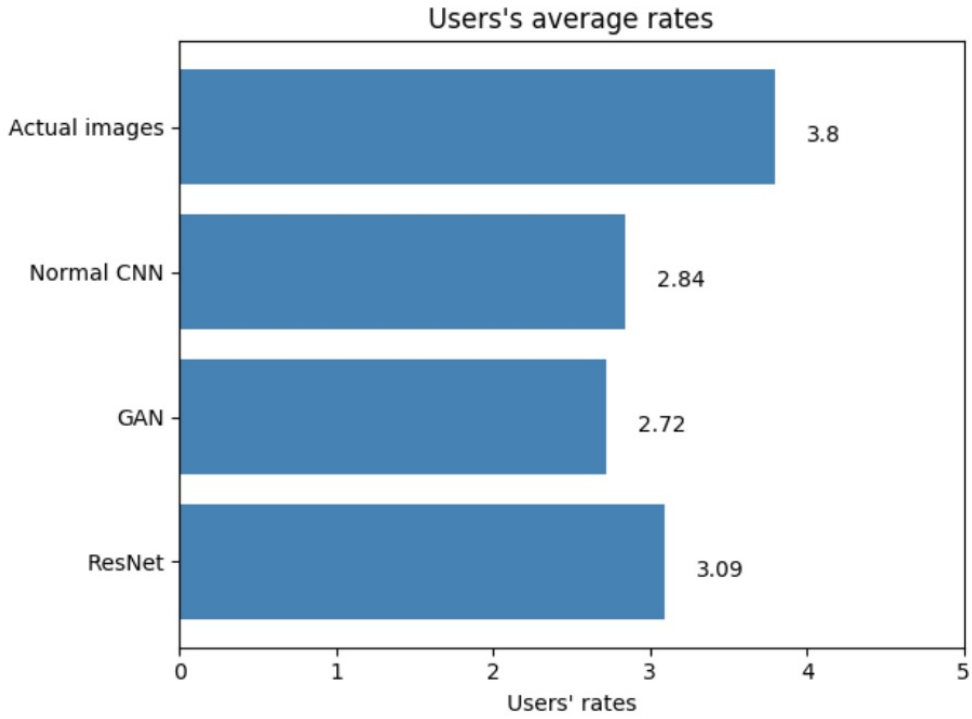


Figure 17: Real images get the highest average rate at 3.8 while ResNet got second at 3.09. GAN has the lowest average rate at 2.72. Generally speaking, three approaches are all providing colorized images with an average rate around three.

# 7 Conclusion and Future Work

In conclusion, deep-colorization is a new rising subject which could provide new dimensions and thoughts to deep-learning as well as image processing. By applying and adding different additional techniques and information, researchers are able to generate vivid or close-to-natural colors for grayscale images.

Three attempts in this report demonstrate the basic algorithms, training strategies, and results of different approaches. It is clear that although some of the results are quite promising and look similar to real colorful ones, there are still many flaws. Firstly, all these attempts require lots of times and computing powers. Secondly, it is hard to pick up correct colors when the images contain too many different textures. Lastly, colorizing performance on a certain image is not predictable and in personal experience, these algorithms seldom create both saturated and accurate colors at the same time.

Additionally, GAN's results are not very promising and definitely did not reach its potential. Further work includes changing parameters in GAN's approach and switching its generator. Current GAN's approach uses U-Net architecture as the generator, which in our comparison and user study, results are not accurate and convincing. An interesting idea is, instead of U-Net, employed the ResNet architecture in the second attempt, which brings better outputs in the experiments, to be the generator. Additionally, codes of the third attempt are confusing and hard to modify. Refactoring is quite important if further investigation wants to be done.

At last, other further thoughts of this subject might involve an application which can use a pre-trained model on an iOS application. The idea of the application is that it can capture the user's old pictures at the real-time, then automatically colorize them. Like an AR application. Users, especially old people, can browse their old black and white pictures in real-time generated colors. It could also be interesting to colorize videos by applying these techniques on a time-sequence. In this case, old black and white films or sports recording can be shown in color mode as well.

# References


[1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015780

[2] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu, "An adaptive edge detection based colorization algorithm and its applications," in *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 2005, pp. 351–354.

[3] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE transactions on image processing*, vol. 15, no. 5, pp. 1120–1129, 2006.

[4] R. K. Gupta, A. Y.-S. Chia, D. Rajan, E. S. Ng, and H. Zhiyong, "Image colorization using similar images," in *Proceedings of the 20th ACM International Conference on Multimedia*, ser. MM '12. New York, NY, USA: ACM, 2012, pp. 369–378. [Online]. Available: http://doi.acm.org/10.1145/2393347.2393402

[5] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 277–280.

[6] A. Y.-S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin, "Semantic colorization with internet images," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 156:1–156:8, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2070781.2024190

[7] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[8] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015, published online 2014; based on TR arXiv:1404.7828 [cs.NE].

[9] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," *CoRR*, vol. abs/1603.08511, 2016, code: https://github.com/richzhang/colorization. [Online]. Available: http://arxiv.org/abs/1603.08511

[10] R. Zhang, J. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *CoRR*, vol. abs/1705.02999, 2017, code: https://richzhang.github.io/ideepcolor/ and https://github.com/junyanz/interactive-deep-colorization. [Online]. Available: http://arxiv.org/abs/1705.02999

[11] F. Baldassarre, D. G. Morín, and L. Rodés-Guirao, "Deep koalarization: Image colorization using cnns and inception-resnet-v2," *CoRR*, vol. abs/1712.03400, 2017. [Online]. Available: http://arxiv.org/abs/1712.03400

[12] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 110:1–110:11, Jul. 2016. [Online]. Available: http://doi.acm.org/10.1145/2897824.2925974

[13] A. Krizhevsky, V. Nair, and G. Hinton. (2009) Cifar-10 dataset. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[14] 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.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] Y. Qu, T.-T. Wong, and P.-A. Heng, "Manga colorization," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 1214–1220.

[17] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proceedings of the 18th Eurographics conference on Rendering Techniques*. Eurographics Association, 2007, pp. 309–320.

[18] G. Charpiat, M. Hofmann, and B. Schölkopf, "Automatic image colorization via multimodal predictions," in *European conference on computer vision*. Springer, 2008, pp. 126–139.

[19] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng, "Intrinsic colorization," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5. ACM, 2008, p. 152.

[20] K. Nazeri, E. Ng, and M. Ebrahimi, "Image colorization using generative adversarial networks," in *International Conference on Articulated Motion and Deformable Objects*. Springer, 2018, pp. 85–94.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[24] E. Wallner. (2017) How to colorize black & white photos with just 100 lines of neural network code. [Online]. Available: https://medium.freecodecamp.org/colorize-b-w-photos-with-a-100-line-neural-network-53d9b4449f8d

[25] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[26] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.