Grayscale Image Colorization In Machine Learning

Preprint · January 2024

DOI: 10.13140/RG.2.2.36374.11849

Zhou Zhou
New York University

Grayscale Image Colorization In Machine Learning

Zhou Zhou
New York University Courant Institute of Mathematical Sciences
251 Mercer Street, New York, NY 10012-1185
zz3382@nyu.edu

Yunqing Zhu
New York University Courant Institute of Mathematical Sciences
251 Mercer Street, New York, NY 10012-1185
yz9661@nyu.edu

Abstract

This research discusses traditional machine learning models, the U-Net model and the GAN model in image colorization. Image colorization focuses on predicting and filling in the missing colors of grayscale images to produce realistic and reasonable color images. This task can improve the interpretability and aesthetic appeal of historical photographs. We use the Kaggle Image Colorization Dataset to train different models, including linear regression, gradient boosting, random forest regression, decision tree regression, the U-net model and a Generative Adversarial Network (GAN) model. It insights into the effectiveness of different machine learning methods in image colorization. Our experiment results indicate the traditional machine learning methods predominantly added a single brown color to grayscale images and the U-net model can generate limited color. In contrast, the GAN model demonstrated better performance in generating more realistic and reasonable color images. It can help to restore and enhance the greyscale images. In future work, we also plan to use the stable diffusion model and large dataset to improve image colorization performance.

1 Introduction

The world's oldest surviving photograph was captured in 1826 or 1827 [1]. The first color photograph was produced in 1861 [2]. However, color photography did not become the dominant form of photography until the 1970s [3]. Many important greyscale images record social activities, art, and daily life at that time. We try to use image colorization to add color to these gray images. This task includes predicting the grayscale image's missing colors and filling in colors at each pixel to generate realistic and reasonably colorful images. These colorful images should be realistic and reasonable. In this process, this process does not need artists to manually add color to these gray-scale images. Besides, image colorization is a very useful technology in the current society. First, people can use this technique to explore historical gray images. It can make history more relatable and engage generations. These revitalized images can help people for education and entertainment purposes. Then, people can use image colorization to upgrade old movies and images. It is more accessible to a contemporary audience accustomed to color content. Finally, image colorization also can improve other fields of work, for example, remote sensing and satellite imaging. Image colorization can enhance interpretation ability and analysis.

Nowadays, people will access many colorful images in daily life. When we try to explore past daily life, these gray images provide a bridge to achieve. Image colorization will transfer these grayscale images to colorful images. These images will become easier to understand and read. In this process, image colorization will improve the accuracy, efficiency, and realism of colorful images. Social activities can use image colorization to help their decision-making and storytelling. Thus, improving image colorization techniques is instrumental for cultural preservation, enhanced visual experience, and advancement of visual technologies.

We do not plan to gather our data to build a dataset, so we will use some datasets to train modules, for example, in Kaggle Image Colorization Dataset [4], there are 5,000 train images and 739 test images. Each image's height and width is 400 pixels. These images contain different daily life scenes, for example, food, sports, animals, and people. In our experiments, we use traditional machine learning methods and neural network methods to implement greyscale image colorization. In this process, techniques used include linear regression, gradient boosting, random forest regression, decision tree, the U-net model, and generative adversarial network to compare their result. We find the traditional machine-learning method will colorize grayscale images to the same color image. The U-net model only can generate a limited image without a discriminator. Besides, the generative adversarial network can generate better color images than the traditional machine-learning method. We use the U-net model as a generator and use a binary classification convolutional neural network as a discriminator. They sometimes can generate an amazing color image but they are not yet reasonable at colorizing complex images.

2 Related Work

Richard Zhang designed a system that is implemented as a feed-forward pass in a Convolutional Neural Network (CNN) at test time and was trained on over a million color images [5]. They use the ImageNet dataset [6] and the SUN dataset [7]. F. M. Carlucci proposes a deep network architecture, exploiting the residual paradigm, that learns how to map depth data to three channel images [8]. They use some public datasets, for example, the Washington dataset [9], the BigBIRD dataset [10], and the JHUIT-50 dataset [11]. Zezhou Cheng developed an adaptive image clustering technique to incorporate the global image information[12].

They both use millions of images to train the model and use deeper CNN to construct their model. In contrast, our model is trained on only 5,000 images, so the dataset scale impacts our model performance compared to their more extensively trained models. We use the traditional machine learning method to implement colorized images. They include linear regression, gradient boosting, random forest regression, and decision tree with MSE loss, They generated almost the same color image. We also explored the GAN (Generative adversarial network) model and U-net model to generate color images. this method is different than Zhang's method in papers. On the other hand, our approach to image format is somewhat similar. We use LAB format to describe an image rather than RGB. Indeed, red, green, and blue channels can consist of a colorful image. RGB image format uses 3 channels to compose a color image, so converting a grayscale image to RGB requires predicting all three channels. By using the LAB format, we just need to predict 2 channels of data from a greyscale image. Because the L channel is the same as the grayscale image channel in RGB format. The LAB is a conversion of the same information to a lightness component L*, and two color components - A* and B*.

3 Method

Our approach involves utilizing images represented in the CIELAB color space format, L* representing lightness, a* representing greenness to redness, and b* representing blueness to yellowness. The L* value has a range from 0 to 100 and a* and b* have arranged from -128 to +127. In this format, the L* value of the image corresponds to the brightness, and the a*, and b* values correspond to color. Our research focuses on predicting pixels with unknown a* and b* values as pixels with unknown color, based on pixels with known color. This model can simulate the image restoration process from noise, and if all the pixels are unknown, the program can be viewed as image colorization from black and white photos. In the research, we test different approaches to predict the pixel's color, including different statistical models of regression analysis, linear regression, gradient boosting, random forest regression, decision tree regression, and the U-net model as a baseline. As a comparison, we use the GAN model as an improved method. We use the mean squared error to measure the performance of each model. In detail, we calculate the average squared error of a*, and b* values in each pixel that needs to be predicted as the error of the whole image. By using this method, the errors from the experiment with different blurring rates are normalized. The MSE loss can be described as $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Furthermore, we used min batch to train every model with a batch size of 32.

3.1 Linear regression

In this approach, we try to estimate the color of one pixel based on its neighbors. The model uses the eight pixels around the central pixel in a three-by-three matrix as input to predict the color of the central pixel. For each pixel, the input size includes the L*, a*, and b* values from the eight surrounding pixels, as well as the L* value from the central pixel, with a total of 25 values. The output size is a*, and b* values from the central pixel, with 2 values. This model can capture the features around the central pixel and use those features to predict the color of given pixel.

In the training process, the goal is to find a linear regression estimator that can accurately predict the color of the target pixel. We tested different regression estimators including OLS Regression, Lasso Regression, Ridge Regression, and Elastic Net. For every image with dimensions N by N, there exist $(n-1)(n-1)$ input data points, excluding those on the edges.

Each unknown pixel has a linear relationship with all neighbors, so the whole image is a large linear system in the prediction process after measuring the coefficients. To accurately calculate the color of each pixel, we need to solve an extensive system of linear equations. In system $Ax = b$, matrix A is a very large sparse matrix, so we use some numerical methods to solve the system to reduce computational complexity. First, we reduce the size of the system by splitting the large image into several smaller parts and solving them separately. For example, the original image has a width and height of 400 pixels. We can split the original image into 16 smaller images with a width and height of 100 pixels each and combine the results after solving them separately. This will reduce the accuracy of the result, especially for those pixels near the edge, but because the color of one pixel is only related to neighbors, most pixels are not affected and the result can still be accurate. Then, because the density of matrix A is low, lower than 6.25%, we use the biconjugate gradient method to approximate the solution to the system of linear equations with a sparse matrix. In this process, four white edges with all zero are added to each part to help predict the pixels on the edge. This will marginally reduce the accuracy.

3.2 Gradient boosting

In the gradient boosting method, we trained 20 models from all training sets. Specifically, we trained 10 models for channel A and 10 models for channel B. In this case, each model is designed to process 400 pixels as input and similarly output 400 pixels. This setup allows for the combination of all output pixels to reconstruct a colorful image. The model update at each iteration t is given by $F_t(x) = F_{t-1}(x) + \alpha h_t(x)$. Here, $F_t(x)$ is the model at the iteration of t, $F_{t-1}(x)$ is the model from the previous iteration, α is the learning rate, and $h_t(x)$ is the new tree.

3.3 Random forest regression

In the random forest regression, we trained 800 models from all training sets. we trained 400 models for channel A and trained 400 models for channel B. We use a different training method than gradient boosting. Every random forest regression model will accept 400-pixel parameters and output 400-pixel output but every model only focuses on predicting a single row of pixels in an image. By combining these predicted outputs with the original L channel, a color image can be reconstructed. Then we can generate a color image. For N trees in the forest and each tree gives a prediction $y_i(x)$ for an input x, then we can write the random forest regression $\hat{y}(x) = \frac{1}{N}\sum_{i=1}^{N} y_i(x)$

3.4 Decision tree regression

In the decision tree, we also trained 800 models like random forest regression. we also trained 400 models for channel A and trained 400 models for channel B. In this case, decision tree regression also will accept 400-pixel parameters and output 400-pixel output. Besides, every model only focuses on its row. Then we can combine the predicted output and the original L channel to generate a color image. Decision tree regression uses variance reduction as a measure of node impurity. The decision tree algorithm will try to partition the data in a way that minimizes the variance in each node. The variance can be describe as Var $= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$

3.5 GAN model and U-net model

Goodfellow and his colleagues introduced a generative adversarial network in 2014 [13]. The GAN model is composed of two neural networks, the generator and the discriminator, which are trained simultaneously. We explored this advanced method to implement image colorization. In the initial experiment, we just used the U-net model to generate color images without the discriminator. Furthermore, we use a U-net model as a generator and use a binary classification network as a discriminator. U-Net is a convolutional neural network that was developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg [14]. Figure 1 indicates the original U-net model. However, we added batch normalization to the U-net model and modified the original U-net model to be capable of our training image size. U-net plays a generator role in the GAN model and U-net can accept the L channel as a parameter and output the A channel and B channel. In this case, the U-net model can predict the AB channel from the L channel. we also built a CNN model as a discriminator. It just needs to classify whether an image is real or fake, so it is a binary classifier. In the training process, the generator (U-net) will learn to produce more and more realistic data over time, so it tries to fool the discriminator. However, the discriminator will learn to distinguish between real and generated (fake) data. The Figure 2 indicates the discriminator architecture. we use batch normalization, Leaky Relu, and dropout in the discriminator model. Leaky Relu can be written as $LeakyReLU(x) = max(0, x) + 0.1 * min(0, x)$. The objective function is defined by a min-max game with a value function $V(G, D)$, it is $min_g max_d V(D, G) = E_{x\char`~p_{data}(x)}[logD(x)] + E_{z\char`~p_z(z)}[log(1 - D(G(z)))]$ [13]. The discriminator tries to maximize $logD(x)$ for real data x and $log(1 - D(G(z)))$ for generated data $G(z)$. The generator tries to minimize $log(1 - D(G(z)))$. In this case, the generator tries to produce data that is as realistic as possible to fool the discriminator into thinking that the generated data is real.

In our GAN model, we initialize weights to the generator and discriminator and we also use Adam optimization for both models. we use binary cross entropy (BCE) loss as a metric to detect GAN model performance. BCE loss can be described as $-\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y}) + (1 - y_i)log(1 - \hat{y}))$. In the training process, we use the real images and label them as 1 to train the discriminator. In this step, we want the discriminator can learn the real image features. Here, we use BCE loss to measure the real image loss. Then, we use the generator (U-net model) to produce fake images and label them as 0 to train the discriminator. Here, we use BCE loss to measure the fake image loss. Then we can add these two losses together to get the final discriminator model loss. In this process, we want the discriminator can learn the fake image features. Finally, we use the generator (U-net model) to produce fake images and label them as 1 to train the discriminator. In this process, we want the discriminator that cannot distinguish these fake images. we can use the BCE loss to measure the final generator loss.

4 Dataset

Our dataset comes from the Kaggle website dataset. This dataset includes test-black directory, test-color directory, train-black directory and train-color directory. The test-black directory has 739 greyscale images and the test-color directory has 739 color images. The train-black directory includes 5,000 greyscale images and the train-color directory includes 5,000 color images. Figure 3 indicates parts of the image colorization dataset, it represents eggs, food, stop sign, and vehicle. Every image's width and height are 400 pixels. These pictures contain different scenes, for example, animals, sports, and daily life. These different scenes can help us improve model performance because we do not focus on specific scene colors. If we focus on a specific scene like a kind of flower, the test greyscale images may cause over-fit. In this case, this dataset can help us to explore diverse scenes to generate more realistic color images. Besides, there are different colors for the same objects in this dataset, for example, a white car and a black car, adding to the complexity and richness of the training.

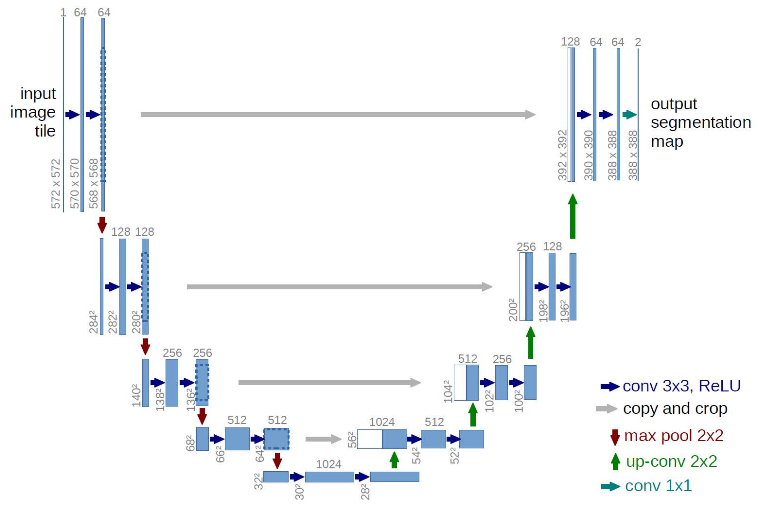

Figure 1: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

[14]

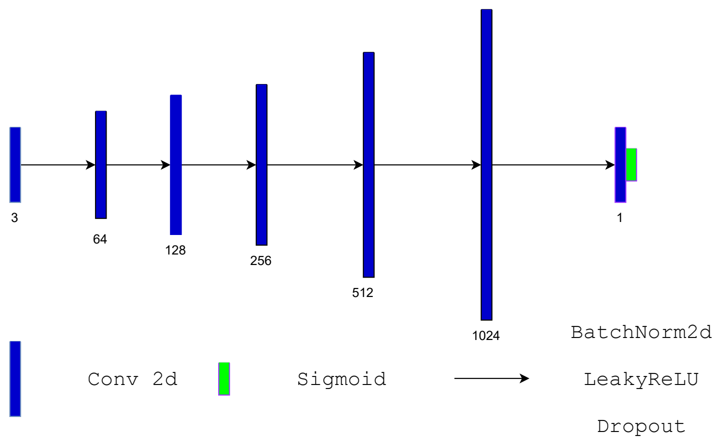

Figure 2: CNN discriminator network architecture

5 Experiments

In this section, we test the performance of different methods of image colorization by numerical experiments. Our objective is to measure their performance under different situations. Through the experiments, we can compare and analyze their performance.

5.1 Linear regression

In the experiment, we use 1000 images with size 400 by 400 to train each estimator and each estimator uses around 150 million data to train. The test size is 100 images. In Ridge Regression and Lasso Regression, we use $alpha = 1.0$ and $l_1 ratio = 0.5$. From the data, we can observe that all linear models can predict the system relatively accurately when the percentage of the unknown pixels is small. However, with the unknown pixels increase, the loss of the predictor drops very fast, showing that the estimators have a poor performance predicting the color based on information other than the color of its neighbors. The behavior of these estimators is more like "paint" inside the region based on given colors introduced by neighbors.

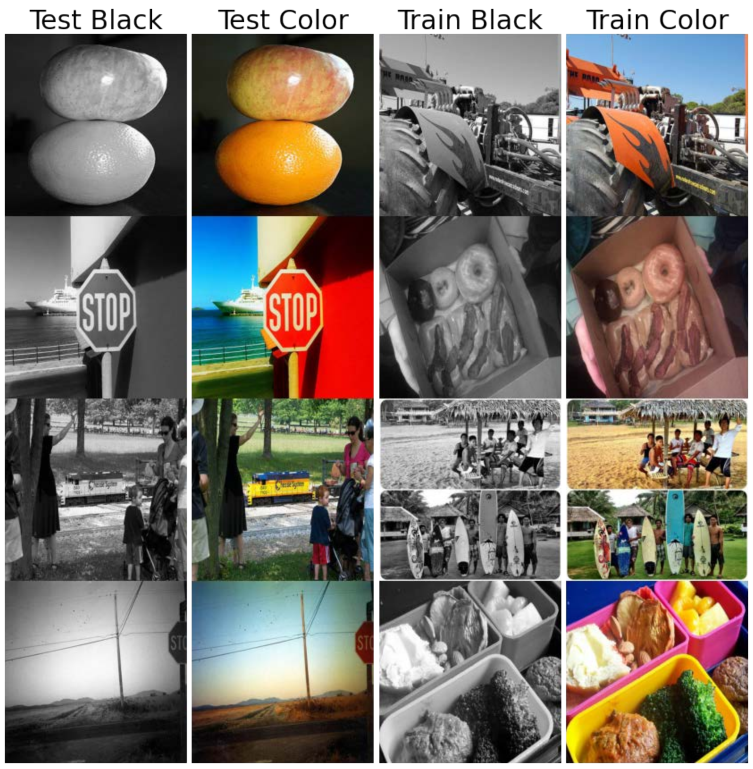

Figure 3: Kaggle image colorization dataset presentation

By comparing the results from different estimators, Lasso regression and ElasticNet regression outperformance OLS regression and Ridge regression by using only parts of parameters instead of all. The L1 regulation helps the estimator deal with highly correlated data. The multicollinearity of parameters affects the performance of estimators as in the model, the color of the eight pixels are correlated with each other. The L1 regulation improves the generalization and therefore improves the estimator performance. Because of the relatively large training set, the effect of L2 regulation is not as powerful as L1 regulation.

Table 1: MES Loss under Different Machine Learning Methods

Percentage of Unknown Pixels	**OLS**	**Ridge**	**Lasso**	**ElasticNet**
20	37.25	48.47	20.12	17.00
40	73.89	69.87	23.89	23.20
60	125.48	102.89	41.52	37.61
80	195.40	136.74	122.59	89.73
100	211.10	148.67	127.55	101.87

Notes: When the losses are large, especially larger than 100, the human eye may not discern differences in the results as the pixels are noise colors visually.

5.2 Other regression model

In the decision tree regression model, the maximum depth is set to 3, representing a balanced value between overfitting and underfitting.

In the gradient boosting model, we use 100 decision trees with a learning rate of 0.1 and a maximum depth of 3. The quality of each split is measured by mean squared error.

In the random forest regression model, we use 100 decision trees with each tree having a maximum depth of 3.

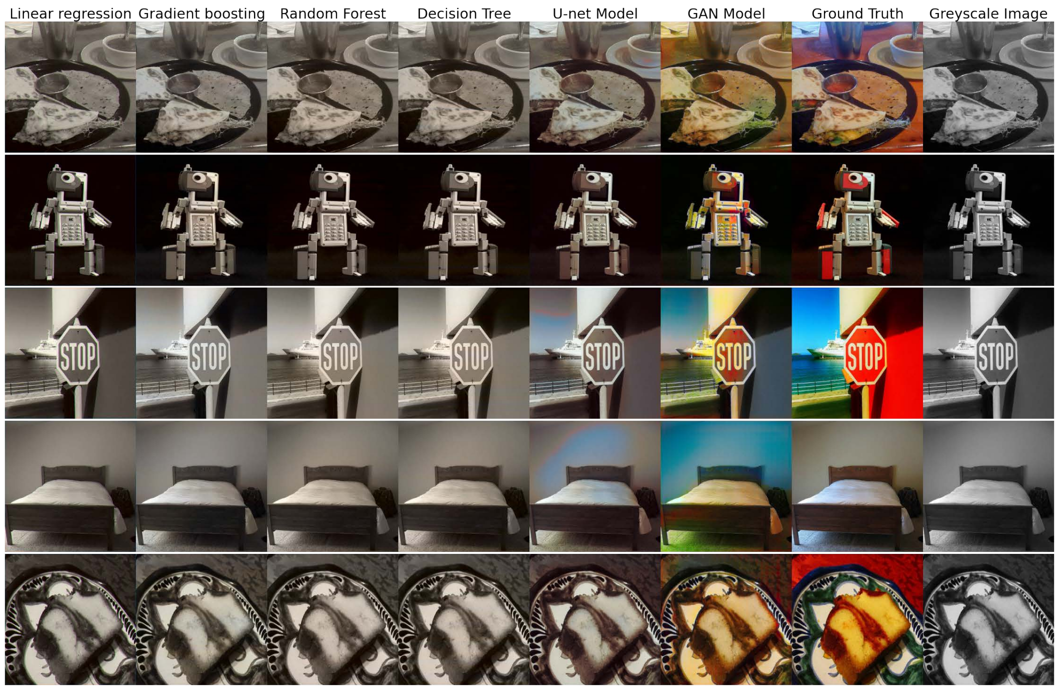

Figure 4: Traditional machine learning method VS GAN model

5.3 GAN model and U-net model

In the GAN model, we set the initial weight to the generator and the discriminator. We also initialize the weights of the batch normalization layer using a normal distribution with a mean of 1.0 and a standard deviation of 0.02 and initialize the biases of the batch normalization layer to 0. We also initialize the weights of the convolutional layer using a normal distribution with a mean of 0.0 and a standard deviation of 0.02. For the GAN model optimizer, we choose the Adam optimizer for the generator model and discriminator model. We set the Adam optimizer learning rate as 0.0002 and set the $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

Figure 5 indicates the MSE loss in different methods. We can easily find the GAN model increases more slowly than other methods with the image number increase. Furthermore, we also can find the traditional machine-learning method loss final results are very close. This means these methods will generate similar results. Based on the MSE result and test result, we can find GAN model performance will generate better color images than the traditional machine learning methods. Traditional machine learning methods only generate a brown color image from greyscale images. In this case, these methods only can generate a single color image, for example, a black-white environment to a brown environment. The U-net model only can generate a limited color image without a discriminator. The GAN model can generate complex colorful and diverse color images. In the experiments, there are some limitations in our work. We only spent about 30 hours training the GAN model for 100 epochs using an A100 GPU. And GAN model sometimes cannot learn anything after a long time of training. In this process, we have to fine-tune the hyperparameter in the model and restart the training process of this model. In this case, We can find the GAN model sometimes cannot generate reasonable color images, for example, some colors blend into the background and some objects remain the greyscale color. In this case, the GAN model struggled with color blending and left some objects in grayscale. With more extensive training, we anticipate improved performance from the GAN model. Besides, recent papers used more advanced models, like diffusion models to implement image colorization rather than the GAN model. And they also use millions of images to train their model. So they can generate more realistic color images. In future work improvement, we plan to explore the use of diffusion models and larger datasets to further enhance our image colorization efforts.

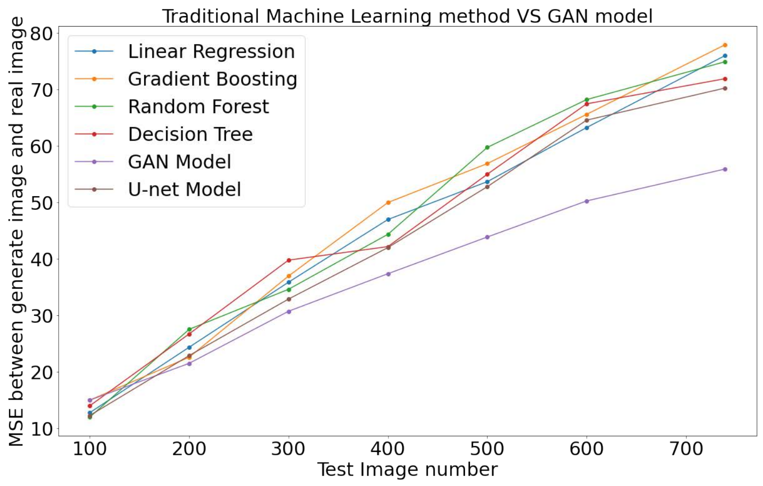

Figure 5: Traditional machine learning method VS GAN model

5.4 Analysis

Using Figure 4 and Figure 5 to compare different models' results, we observed distinct differences in the performance of various models. In this case, we can find the traditional machine learning methods do not generate reasonable color images from greyscale images. These models just add brown color to the greyscale images and show no ability to predict the color of the image. In contrast, the GAN model can generate better color images than traditional machine learning methods.

6 Conclusion

In conclusion, we plan to implement image colorization through linear regression, gradient boosting, random forest regression, decision tree regression, and GAN model. In this process, we trained these models using the Kaggle image colorization dataset. To compare the loss between different methods, we normalize the loss of different methods to get comparable results. When comparing the performance of classical machine learning methods, we only consider the situation that the color of all pixels is unknown because this situation is closer to reality.

Some advanced models like the stable diffusion model from other researchers can generate more realistic color images than our GAN model. However, we use much smaller datasets to generate some reasonable color images. Despite the limitation of using fewer images as training data instead of using millions of them, our GAN model was able to produce reasonably accurate colorization with a limited dataset, especially when the target image does not have complex scenes involving multiple objects. With more prompts, the classical machine learning model can also be used to paint the image with the given color. Besides, there are some defects in the models. The classic machine learning methods are unable to generate high-quality color images without prompts. Also, the GAN model sometimes cannot converge after many training epochs, in this case, we must restart the model training process and fine-tune the hyperparameter.

For the real-world problem, there are many old greyscale images in daily life. People can use our model to implement image colorization instead of painting color manually. Image colorization can help people cognitively and understand society's environment. For theoretical Implications, our GAN model challenges traditional machine learning in image colorization fields but it still has a gap than some advanced generation models. It confirms that using traditional machine learning methods in image colorization is not a proper choice. Our model may instead of manually adding color to greyscale images work. In our research progress, we follow the train, result, and fine-tune loop. We read some papers to implement the U-net model and discriminator model. We also meet some challenges, for example, the GAN model training process is much waste of time, so we only train 100 epochs for the GAN model. Based on our findings, we believe the GAN model will generate better color images than the traditional machine learning method.

References

[1] Megan D Robinson. What you don't know about the world's oldest photograph, 2022. Last accessed 21 October 2023.

[2] The independent. From charles mackintosh's waterproof to dolly the sheep: 43 innovations scotland has given the world, 2016. Last accessed 21 October 2023.

[3] Peerspace. When was color photography invented?, 2022. Last accessed 21 October 2023.

[4] AAYUSH SHARMA. Image colorization dataset, 2021. Last accessed 26 October 2023.

[5] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[7] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.

[8] F. M. Carlucci, P. Russo, and B. Caputo. (de co: Deep depth colorization, 2018.

[9] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE International Conference on Robotics and Automation*, pages 1817–1824, 2011.

[10] Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516, 2014.

[11] Chi Li, Austin Reiter, and Gregory D. Hager. Beyond spatial pooling: Fine-grained representation learning in multiple domains. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4913–4922, 2015.

[12] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization, 2016.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[14] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).