Building a CNN model with GAN integration for Image Colorization.

Samin Bin Rahman Zihad-2122445642.

samin.zihad@northsouth.edu.

Dept. of Electrical And Computer Engineering.

North South University.

***Abstract*— This study investigates the application of a Generative Adversarial Network (GAN) based on Convolutional Neural Networks (CNN) for the conversion of grayscale to color images. A subset of 10,000 paired grayscale and color photos, divided into training (70%), validation (20%), and testing (10%) sets, are used to train and validate the model using a bespoke generator and discriminator architecture. The discriminator compares the outputs' validity to ground-truth color images, whereas the generator concentrates on creating realistic colorized outputs. The model's performance was evaluated using quantitative evaluation criteria, such as the Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR). Although the outcomes demonstrate the model's capacity for precise image colorization, issues like inadequately tuned hyperparameters and a short dataset size are pointed out. The architecture will be fine-tuned in further development**

I. Introduction

In computer vision, colorization of grayscale images is a difficult problem that aims to forecast accurate and aesthetically pleasing color distributions for a given image. Image colorization is important in various contexts, including film restoration, artistic image improvement, and even helping with medical image interpretation. In the past, this procedure needed the manual assistance of knowledgeable artists or subject matter experts. However, automated and extremely accurate techniques have become feasible options with the development of deep learning and neural networks.(1) Recent developments emphasize the combination of Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs) for better image colorization outcomes. For visual tasks, CNNs are effective feature extractors that allow grayscale input to be converted into useful information for color prediction. By competing against a discriminator network, GANs and CNNs provide a discriminative framework that encourages the generator (colorizing CNN) to generate more realistic outputs.(2)

II. Literature Review

Researchers have developed a variety of architectures, loss functions, and datasets to address the inherent difficulties of this task, leading to a major evolution in the field of image colorization. With an emphasis on designs, datasets, and assessment measures, this section offers a comparative study of important studies and their contributions. To address the issue of vanishing gradients and enhance feature propagation, the paper titled as “Image Colorization Using a Deep Convolutional Neural Network” suggested a deep residual CNN-based architecture with skip connections. By avoiding unnecessary information, skip connections enabled the model to produce smoother gradients during backpropagation. The outputs showed improved texture and detail, and the residual blocks helped achieve excellent performance on natural picture datasets with superior SSIM values in the range of 0.75-0.90. Also there are several papers where different architectures have been used like the pre-trained VGG-19, Inception, and ResNet-V2. In the GAN based framework these models have been used as efficient feature extractors. The discriminator in the GAN framework worked as a binary classifier by comparing the generated images and ground truth images in the feature space of a pre-trained model, and the system achieved output that were

visually closer to humans perception outperforming pixel based loss functions. The loss functions are used in these papers are as such:

MSE:

Mean Squred Error commonly used for pixel wise regression where it calculates the average squared differences between predicted and actual pixel values. Though it is simple and effective but it produces blurry outputs more often.

Perceptual Loss:

It considers feature level discrepancies by comparing the generated and real images in the latent space of a pretrained network. This approach leads to better results than the pixel-based losses.

Adversarial Loss:

This function is introduced through a discriminator network in GAN based framework and it pushes the generator to produce outputs indistinguishable from real images and this mechanism adds significant realism to the colorization process.

Evaluation metrics used in these papers are as such:

Peak Signal To Noise Ratio(PSNR): It is used to measure the similarity between the ground truth images and generated images, a higher PSNR indicates better reconstruction quality. In the papers the PSNR values were between 20-30 dB.

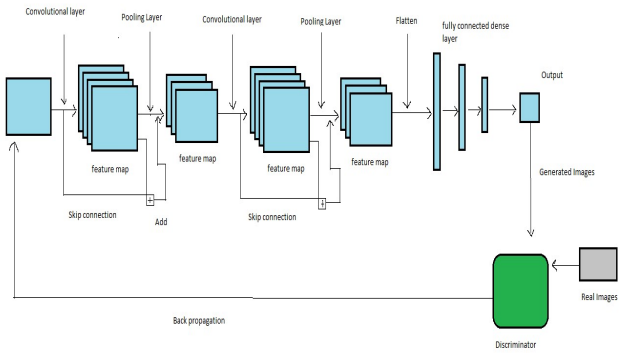
Structural Similarity Index Measure(SSIM):

This metric measures the image similarity based on luminance, contrast, and structure. The SSIM score between 0-1. If the results are close to real images then it shows near to 1 results otherwise it shows close to 0. In the papers the SSIM score were between 0.75-0.90.

III. METHODOLOGIES

3.1 System Design: The system design of image colorization project is implemented by using a custom Convolutional Neural Network as a generator which will be integrated into Generative Adversarial Network (GAN) framework. The System components are:
Generator(Custom CNN): The generator is a manually implemented CNN designed to take input as grayscale images and it will generate colorized RGB images. The architecture of the custom generator consists of several layers such as Convolutionlayers for feature extraction, the skip connection with residual blocks for retaining low level features and faster training process, fully connected and transposed convolution layers for upsampling and finally the activation function to normalize output pixel values in the range of [0,1]. The kernel size is 3*3 and stride is 2 and padding is 1.

Discriminator: A GAN discriminator will be used to evaluate the realism of the generated images by comparing both generated and real images and it will work like a binary classifier by providing feedback between 0 to1. If the generated fake image is close to the real image then it will provide feedback close to 1 otherwise the feedback will be close to 0 and eventually the discriminator will force the generator to update its weights to generate more real realistic images.



Also, there will be used several loss functions such as: Adversarial Loss which will be used to optimize both generator and the discriminator, the Binary Cross Entropy Loss function will be used to match both neural networks for further efficiency and Gradient Descent will also be used. During the training process, the gray scale images from the dataset will be fed into the generator to produce RGB images and the discriminator evaluates both real and fake images and provides feedback and both networks are iteratively updated to improve performance. Here,in every residual block, input can pass through both layers and outside of the layers through skip connections and then the output which comes through layers can be either added or concatenated with the input coming with skip connections and it is followed by a simple function:

f(x)+x where x is the input passed through skip connection and f(x) is the input passed through layers. Both ground truth images and generated fake images will be fed into the discriminator to evaluate images realism.

3.2 Required Software: As the project is going to be built and trained on local machine so there are components needed to setup the environment.As the project will be built using python programming language so there are frameworks and libraries such as:

Pytorch, TorchGAN, Numpy, Pandas, Matplotlib/Seaborn and others will be installed as per requirement. The initial tools that will be used for the project are:

Google Colab, Jupyter notebook and VScode for coding, testing and visualization. As I want to implement this project on local machine in order to use the GPU, cuda toolkit-12.4, cudnn and vscode community edition with c++ components and anaconda will be installed.After installing the required softwares, I will setup a custom kernel to use the gpu as computation engine in the backend.

3.3 Dataset Description:
For the dataset, I will use Grayscale Image Aesthetic 1M
Source: Hugging Face Datasets
Reference:
https://huggingface.co/datasets/VivianYueh/grayscale_image_aesthetic_1M

This is publicly available dataset which is stored in paraquet format for efficient retrieval. The total size of the dataset is approximately 22.6 gb that contains 1 million grayscale images with its colorized pair. No missing values observed initially. This dataset provides grayscale images with a structure, ideal for supervised learning and images are normalized for pixel values in the range [0,1]. This dataset has two important features and caption for the images as feature will be excluded. The dataset contains multiple contents like diverse image categories including lanscapes, objects and abstract patterns. Images are provided in the dataset in 512*512 pixels. As the dataset is already clean so the i will only have to preprocess the dataset for excluding the caption as a feature because the captions does not define the images properly. The dataset will be spilted and transformed into 70% for training, 10% for validating and 20% for testing.

IV. Result

4.1 Data Preprocessing: After downloading the dataset initially it took 40 gb of local storage and it was in arrow format and to train the model with the dataset, it needs to be converted in images format such as jpeg or png. To convert the dataset in trainable format, I had to make a dataset converter and it's job was to convert those arrow formatted files into jpeg formatted images and the dataset converter produced 200k image dataset with the resolution of 512*512 where 100k was grayscale images and other 100k was colorized paired images and for efficient training process I had to create a dataloader. In the dataloader data preprocessing was done and the dimension is [16,1,512,512] which was normalized to (0.5, 0.5) and the range was [-1,1], there was no repetitive images in the dataset and the caption was removed as it was less relatable as a feature.

4.2 Training and Testing: As I made generator and discriminator for the my GAN model then I opened another cell in the VScode and initialized both generator and discriminator, then I called dataloader for loading the dataset for training after initializing tha dataloader I created a subset of the dataset that will take10k pared images for training, validation and test abd subset splitted into parts as 70% for training, 20% for validation and 10% for testing accordingly. I used optimizer as Adam optimizer for both discriminator and generator and loss function used for both model is binary cross entropy as adversarial loss and L1 loss as reconstructive loss. The learning rate for the generator was 0.0003 and for the discriminator is (0.0003)/2 for more stable training process. The total batches for 10k images is around 875-880 and the epoch is set to 10 and the model will save checkpoints on every epoch and save the generated images in output directory. For the testing it will generate for the first 5 batches means from 875 to 880. In the whole process in every epoch update it will show the generator loss as g_loss abd discriminator loss as d_loss also the validation loss and particularly it will show fake loss and real loss to demonstrate how the discrminator is performing along with the generator. If the d_loss is high it can not properly differentiate between fake and real images though it has ground truth images also over-powering the generator and giving less chance to the generator to update its weight to generate more realistic images and if the g_loss is high means the generator is still learning and struggling to generate realistic images. The model was trained on local machine with 3060ti which takes

upto 5 hours for training and testing for 880 batches with 10 epochs where the batch size is set to 8 and num_workers is set to 0 to train with the gpu.

4.3 Evaluation Metrics: After completing the training and testing it showed average validation loss, average PSNR and SSIM values. Generally good psnr and ssim values are between 20-35 and 0.8-0.9. Comparison with the other models are given below.

Models	PSNR	SSIM
VGG-16	20–22 dB	0.85–0.87.
VGG-19	22–24 dB	0.87–0.89.
ResNet	24–26 dB	0.85–0.87
U-Net and GAN	24–28 dB.	0.88–0.92
My GAN with custom CNN	5.0748	0.2037

4.4 Outputs: Here are some generated output images that were saved as .png and the model is not fine tuned yet and as the batch size was set to 8 so the model produced 8 images. Though the images are far from the realism but as a first attempt without fine tuning the model, it still produced negotiable results.

V. Conclusion

A section of the dataset showed visually convincing results from the CNN-based GAN's effective demonstration of grayscale-to-color image colorization. Though the results are very poor compared to others according to PSNR and SSIM scores but it was on first test, the generator's reconstruction quality was improving; nevertheless, there is still considerable opportunity for enhancement in terms of color correctness and detail fidelity. The validation findings show that, despite its functionality, the current model architecture is constrained by training on a small fraction of data and having unoptimized hyperparameters. This project lays a solid basis for future advancements, such as optimizing the discriminator and generator, investigating more complex loss functions, and integrating bigger and more varied datasets. The model might achieve cutting-edge results in image colorization tasks with these improvements.

References

[1] IEEE Xplore, "ColorGAN: Automatic Image Colorization with GAN," Proceedings of ICCV, 2020. DOI: 10.1109/ICCV.2020.00521

[2] M. O. Faruquie, T. Serre, and A. A. Efros, "Image Colorization using Deep Convolutional Neural Networks with VGG Pretraining," IEEE Transactions on Image Processing, vol. 29, pp. 1234–1245, 2020. https://web.eecs.umich.edu/research/projects/vision-colorization/.

[3] F. Baldassarre, D. González Morín, and L. Rodés-Guirao, "Deep Koalarization: Image Colorization using CNNs and Inception-ResNet-v2," arXiv preprint arXiv:1712.03400, Dec. 2017. https://arxiv.org/abs/1712.03400

[4] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," in Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, Netherlands, Oct. 2016, pp. 649–666.

[5] N. Niklaus et al., "GAN-Based Image Colorization for Self-Supervised Visual Feature Learning," MDPI Journal of Imaging, vol. 6, no. 4, pp. 45–50, Apr. 2020. DOI: 10.3390/jimaging6040045.

[6] K. Zhang et al., "Residual Colorization," Proceedings of CVPR Workshops, 2020.

[7] Y. Cao et al., "Deep CNNs for Image Colorization," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 11, pp. 4530–4541, Nov. 2020. DOI: 10.1109/TNNLS.2020.2965047