**GREYSCALE IMAGE COLORIZATION USING GAN’s**

By: Uma Prabhath Reddy

**Abstract:**

The task of colorizing grayscale images is a challenging yet impactful problem with significant applications in fields such as digital restoration, historical preservation, and computer vision. This project proposes a novel approach to grayscale image colorization using a Conditional Generative Adversarial Network (CGAN). Unlike traditional methods that require manually crafted features or external information, our model directly learns the mapping from grayscale input images to their corresponding colored outputs. The input consists of single-channel grayscale images of fixed size (32x32 pixels), while the output is a full-color version of the same image, maintaining spatial dimensions.

The network is trained on publicly available datasets, including CIFAR-10, enabling the model to learn color patterns across diverse categories such as animals, vehicles, and everyday scenes. The conditional architecture of the CGAN allows the generator to produce more accurate and realistic colorizations by conditioning on the input image, leading to visually plausible results across various image types.

To train the model effectively, we employ data augmentation strategies to increase the diversity of the grayscale inputs and prevent overfitting. The adversarial loss from the discriminator helps refine the generated colorization, improving the realism of the output. Experimental results demonstrate the network's ability to generate convincing colorized images, although challenges remain in accurately capturing color details, especially in more complex or ambiguous image regions.

This work presents a promising step forward in fully automated image colorization using deep learning techniques, offering potential for applications in multiple domains, including the restoration of old photographs, enhancement of scientific imaging, and more.

**1. Introduction**

**1.1 General**

Image colorization is a complex and intriguing problem with wide-ranging applications, including digital restoration, historical photo enhancement, and visual content creation. Transforming grayscale images into color not only enhances their visual appeal but also provides additional context, making historical artifacts more relatable and improving user engagement across multiple domains. Manual colorization, however, is a labor-intensive and subjective task that often requires expertise and significant time. This challenge has spurred interest in automated methods, especially in the field of deep learning, which has significantly advanced the capabilities of image colorization.

In recent years, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have revolutionized image colorization by enabling models to learn complex color patterns from images and generate plausible colors without manual intervention. Unlike traditional methods that rely on user input or hand-crafted features, deep learning models can analyze grayscale images and predict color based on spatial and contextual cues, vastly improving both accuracy and realism. These advancements offer significant potential in automating processes such as art restoration, media enhancement, and user experience optimization in fields like UI/UX design.

**1.2 Objective**

The primary objective of this project is to develop an automated solution for colorizing grayscale images using a Conditional Generative Adversarial Network (CGAN). This model takes a single-channel grayscale image as input and generates a corresponding color image, leveraging the power of GANs to produce realistic and contextually accurate colorization.

The model will be trained on the CIFAR-10 dataset, which contains a diverse set of images, allowing the model to learn various visual patterns and color associations across categories such as animals, vehicles, and everyday scenes. To enhance the model’s robustness and generalization, data augmentation techniques such as rotation, zoom, and horizontal flipping will be employed.

The specific goals of this project is to create a reliable model that can colorize grey scale images with high accuracy. Through qualitative and quantitative evaluations, including visual inspections and Mean Squared Error (MSE) analysis, this project aims to contribute to the growing field of image colorization, offering valuable insights and methods for future applications in restoration, content creation, and design.

**2. Literature Review**

**2.1 GENERAL**

The concept of Generative Adversarial Networks (GANs) has gained significant attention in recent years, primarily due to their ability to generate highly realistic data and images. The foundational work by Ian Good fellow and his collaborators in 2014, which introduced the GAN framework, has been instrumental in understanding the underlying mechanics and potential applications of GANs. To comprehend the basic operation of GANs, the original paper on GANs was reviewed to establish a solid theoretical foundation for the project.

In the context of image colorization, several studies have explored the use of GANs to transform grey scale images into realistic colour representations. Notably, the work by Kamyar Nazeri et al. (2018) on "Image Colorization Using GANs" presents a method that adapts GANs for automatic image colorization. This research provided valuable insights into the architecture and structure of Conditional GANs (cGANs), which can be used to generate images conditioned on specific inputs, such as grayscale images.

The architecture and hyperparameter tuning play a crucial role in the effectiveness of GAN-based models. To improve the performance of our model, various resources, including online blogs and tutorials, were consulted to optimize the network’s architecture and to learn best practices for training GANs efficiently. These resources helped identify key strategies for overcoming challenges such as mode collapse and improving the stability of training.

**2.2 LITERATURE REVIEW**

| **Sr. No.** | **Name of Paper** | **Author(s)** | **Year of Publication** | **Content Referenced** |
| --- | --- | --- | --- | --- |
| 1 | Generative Adversarial Nets | Ian Goodfellow, et al. | 2014 | Concept and internal working of GANs |
| 2 | Deep Learning for Image Colorization | Xie et al. | 2017 | Use of deep learning techniques, particularly CNNs, for improved colorization |
| 3 | A Survey of Generative Models in Image Processing | Li et al. | 2021 | Overview of generative models and their applications in image restoration and manipulation |

### 3. GAN

Generative Adversarial Networks (GANs) were first introduced by Ian Goodfellow in 2014 as a powerful method for generating data. A GAN consists of two key components: a **generator** and a **discriminator**. These two networks compete against each other in a zero-sum game. The **generator** creates new data (like images), while the **discriminator** evaluates whether the data is real (from the training set) or fake (generated by the generator).

**3.1 The Generator**: The generator’s job is to learn how to create data that looks similar to the real data. It starts with random noise and gradually improves through feedback from the discriminator.

* 1. **The Discriminator**: The discriminator is a classifier that learns to distinguish between real data and data generated by the generator. As the generator improves, it becomes harder for the discriminator to tell the difference.

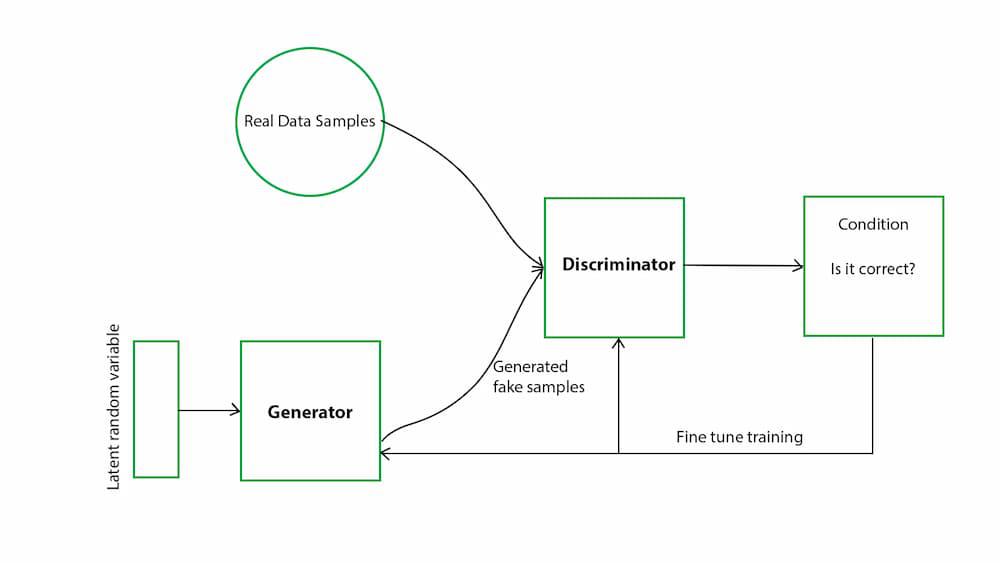


Fig 1: GAN Architecture

**3.3 Working of a GAN**

Both networks improve through continuous feedback: the generator tries to fool the discriminator, while the discriminator tries to get better at catching the generator’s fake data.

In mathematical terms, the generator is trained to minimize the likelihood of the discriminator correctly identifying fake data, and the discriminator is trained to maximize its accuracy in distinguishing real from fake data. This creates a dynamic training process, where both networks push each other to perform better.

#### 3.3.1 Training the Discriminator

The discriminator looks at real images (training samples) and generated images separately. It distinguishes whether the input image to the discriminator is real or generated. The output *D(X)* is the probability that the input *x* is real, i.e. *P(class of input = real data instance).* We train the discriminator just like a deep network classifier. If the input is real, we want *D(x)=1.* If it is generated, it should be zero. Through this process, the discriminator identifies

features that contribute to real data instances.

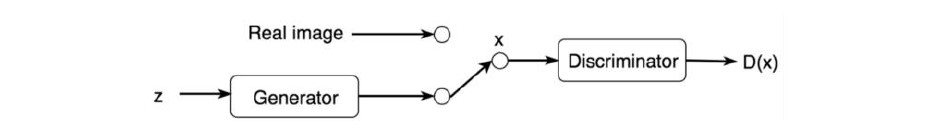


Fig.2 Training Discriminator

The discriminator outputs a value ***D(x)*** indicating the chance that *x* is a real data instance. Our objective is to maximize the chance to recognize real data instance as real and generated data instance as fake. i.e. the maximum likelihood of the observed data. Cross entropy is used for cost option.

#### 3.3.2 Training the Generator

We want the generator to create images with D(x) = 1. So we can train the generator by back propagating this target value all the way back to the generator, i.e. we train the generator to create data instances that are inclined towards what the discriminator thinks it is real..

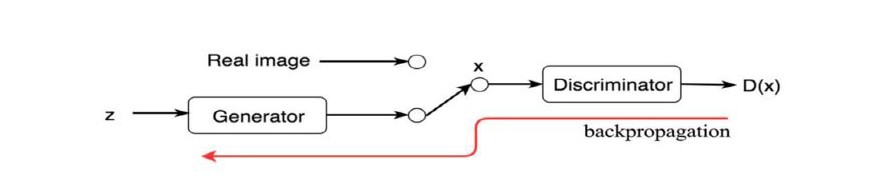


Fig 3: Training Generator

The generator’s training involves backpropagating the error from the discriminator’s output back through the network, encouraging it to improve its ability to mimic real data. Essentially, the generator is adjusted to create data that aligns with the features the discriminator considers real.

On the generator side, its objective function wants the model to generate images with the highest possible value of *D(x)* to fool the discriminator.

#### 3.3.3 Simultaneous Training

Once both objective functions are defined, they are learned jointly by the alternating gradient descent. We fix the generator model’s parameters and perform a single iteration of gradient descent on the discriminator using the real and the generated images. Then we switch sides. Fix the discriminator and train the generator foranother single iteration. We train both networks in alternating steps until the generator produces good quality images.

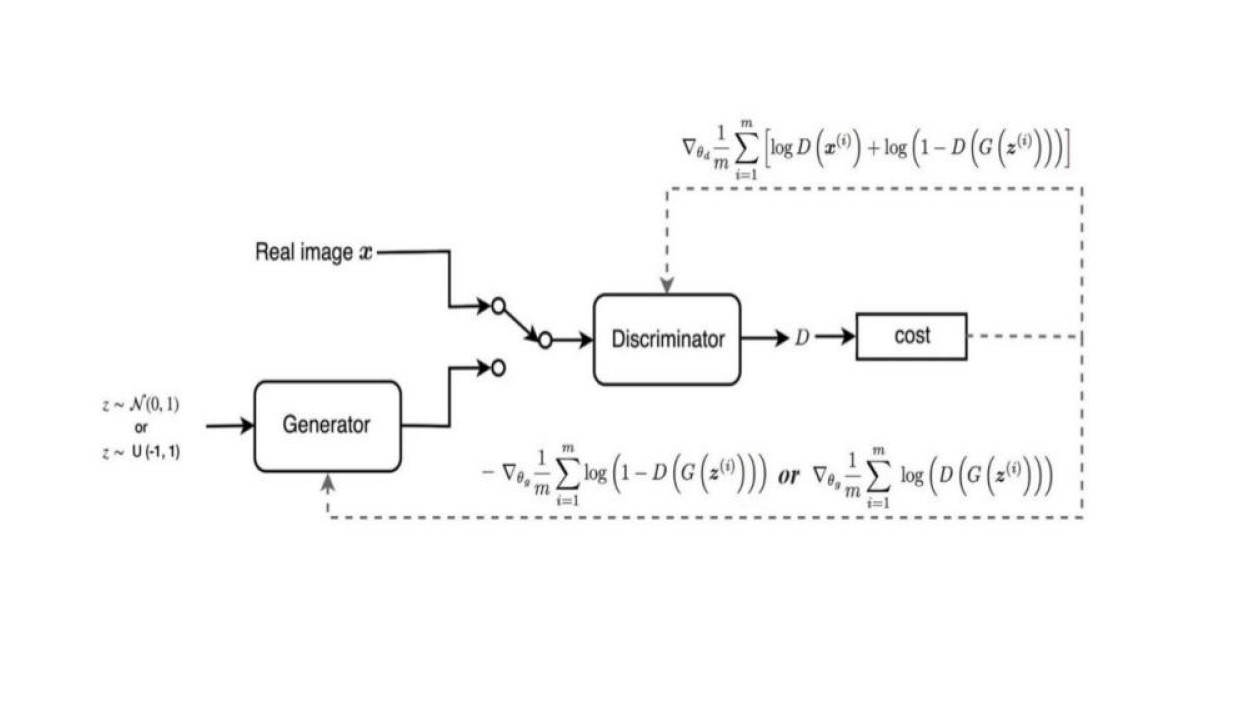


Fig 4: Traning a GAN

**4. GAN for image colorization**

**4.1 Method**

The problem at hand involves Image-to-Image translation, specifically focusing on mapping grayscale images to their corresponding colored versions. This task falls under the category of **regression at the pixel level**, where the model's goal is to predict the color for each pixel in the grayscale image while preserving the spatial structure and integrity of the image. The output must have a **high degree of similarity** to the input, in terms of spatial dimensions, and the color information for each pixel needs to be accurately predicted.

Given that the input image is grayscale and the output is a colored version, the network must maintain a similarity in the **spatial dimensions** between the input and output. The network is designed to process these images at the pixel level, learning the mapping from a **grayscale 32x32 image** to a **colored 32x32 image**, while ensuring the correct colorization for each pixel. The structure of the output image must adhere to the same dimensions, so it maintains its alignment with the input in terms of both spatial properties and pixel-wise information.

### 4.2 GAN Architecture

Our model leverages a **Generative Adversarial Network (GAN)** for the task of **image colorization**, specifically designed as an **Encoder-Decoder** architecture. The architecture consists of two primary components: the **generator** and the **discriminator**.

#### 4.2.1 Generator Architecture:

The **generator** takes in a grayscale image of shape (32 x 32 x 1) as input. The network architecture consists of **convolutional layers**, which help extract features from the input image. Unlike traditional networks with pooling layers, the generator architecture uses **downsampling** to progressively reduce the size of the image, followed by **upsampling** to reconstruct the image at the same dimensions as the original input (i.e., 32x32 pixels).

**Initial Down sampling**:

The input grayscale image is passed through a series of convolutional layers with progressively increasing filter sizes and strides. Initially, a 5x5 kernel with a stride of 1 is used, followed by a series of smaller kernel sizes (3x3) and strides of 1. This helps capture fine details in the image.

**Compression**:

The image is progressively compressed through successive convolutional layers with a **kernel size of 3x3** and a stride of **2**, reducing the image dimensions. After four downsampling stages, the resulting tensor has dimensions of (2 x 2 x 128), effectively reducing the spatial dimensions while increasing the depth of the features.

**Upsampling**:

To reconstruct the image, the compressed representation is then **upsampled** using **transposed convolutions** (also known as **deconvolutions**), with kernel sizes of 3x3 and strides of 2, which helps increase the image dimensions back to (32 x 32 x 3). In this expansion stage, **concatenation** of intermediate layers (from the downsampling path) is done with their corresponding upsampling layers, following the **Encoder-Decoder architecture**. This ensures that fine-grained spatial details are preserved during the upsampling process.

* **Dropout and Batch Normalization**: During upsampling, **dropout** with a rate of 0.5 is applied in the first two layers to introduce noise and enhance the robustness of the generator. **Batch normalization** is applied at each layer to stabilize training and improve convergence by normalizing the activations of each layer.
* **Activation Functions**: In terms of activation functions, **LeakyReLU** with a slope of 0.2 is used throughout the network, as it has shown to outperform the standard ReLU activation in terms of preventing **dead neurons** during training. This non-linearity helps the network learn more complex mappings. For the final output layer, a **tanh** activation function is used, which scales the output values to the range of [-1, 1]. This is particularly useful when working with pixel values, as it matches the expected range for colorization tasks.
* **Output**: The final output is a 32x32 RGB image, where each pixel in the image corresponds to a predicted color value.

#### 4.2.2 Discriminator Architecture:

The **discriminator** works as a binary classifier, distinguishing between **real** and **fake** images. It receives two possible inputs: the **ground truth colored image** and the **generated image** from the generator. These images are concatenated with the original grayscale image on the **channel axis**, forming a colored image (of shape 32x32x3). The discriminator is designed to classify these images as either real (from the dataset) or fake (generated by the generator).

* **Convolutional Layers**: The discriminator uses multiple **convolutional layers** with kernel sizes of 3x3 and strides of 2. The **downsampling process** is performed here, reducing the image size while extracting high-level features. After each convolutional layer, a **LeakyReLU activation function** with a slope of 0.2 is used, helping the network avoid dead neurons and ensuring the learning process remains stable.

**Final Classification Layer**: After the convolutional layers, the output is **flattened**, followed by a fully connected hidden layer with **128 units**. The final output layer consists of a single neuron with a **sigmoid activation function**, providing the **probability** that the input image is real (ground truth) or fake (generated by the generator).

### 4.3 Training Strategy:

The **training objective** for the **generator** is to minimize the difference between the predicted colorized image and the actual ground truth image. This is typically achieved using a **mean squared error (MSE)** loss function that quantifies the pixel-wise differences between the generated and real images.

For the **discriminator**, the objective is to accurately classify real and fake images, which is done using **binary cross-entropy loss**. The discriminator is trained to distinguish between real images and those generated by the generator.

During training, the **generator** and **discriminator** engage in an adversarial process:

* The **generator** learns to create realistic images that can fool the discriminator into classifying them as real.
* The **discriminator** learns to distinguish between real and fake images, providing feedback to the generator.

The model is trained for **100 epochs** with a **batch size of 64**. The **Adam optimizer** is used for both the generator and discriminator with a learning rate of **0.0005**, ensuring stable training and faster convergence.

**4.4 Results**

-The top image shows the single channeled image (Equivalent to GrayScale image)

-The middle image shows the image generated by our GAN

-The bottom image shows the ground truth.

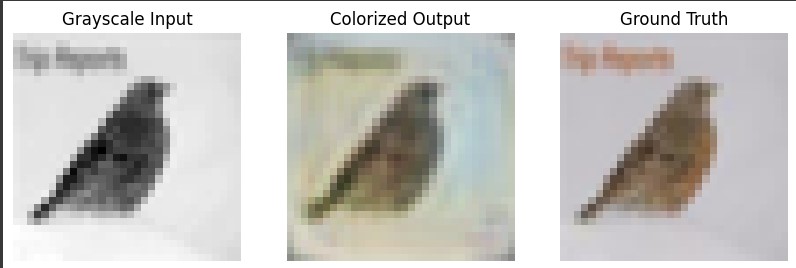


Fig : 5.1

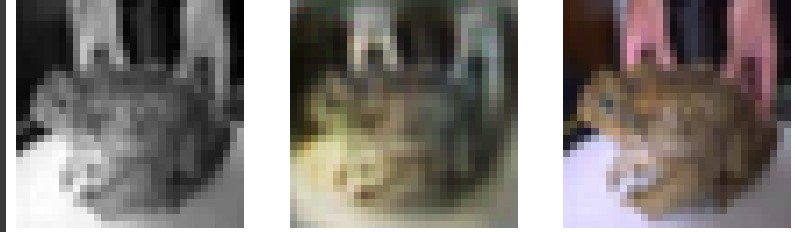


Fig 5.2



Fig 5.3

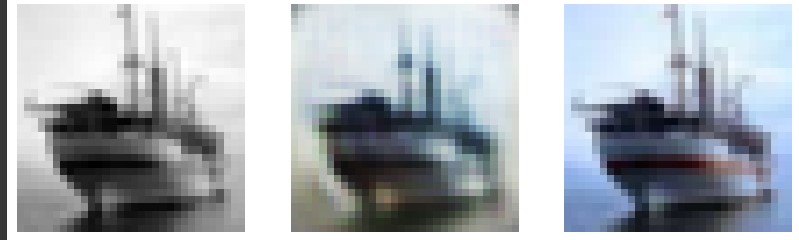


Fig 5.4

### 5. Summary and Conclusion

### 5.1 Summary

This work presented a CGAN-based approach to colorizing grayscale images. We discussed the architecture, training strategies, and evaluation methods, showing that the model can generate colorized images with a reasonable level of realism. The evaluation included both qualitative and quantitative metrics, demonstrating the model's strengths and areas for improvement.

### 5.2 Conclusion

In conclusion, the use of CGANs for image colorization offers promising results, though challenges such as accurately predicting complex colors remain. Future work could focus on enhancing the dataset, incorporating attention mechanisms, or exploring other adversarial training strategies to improve colorization accuracy and realism.

Despite these challenges, the overall performance of the model was promising. With more **training iterations**, **improved architectures**, and better **regularization techniques**, the GAN could potentially overcome these issues and produce even more accurate and realistic colorized images. This project highlights the immense potential of GANs in tasks involving **image generation** and **transformation**, with plenty of room for enhancement through further research and development.

**REFERENCES**

 **Kamyar Nazeri, Eric Ng, and Mehran Ebrahim**. "Image Colorization Using Generative Adversarial Networks." arXiv:1803.05400 [cs.CV], 2018.

 **Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio**. "Generative Adversarial Nets." In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

 **Li et al.** "A Survey of Generative Models in Image Processing." 2021.

 **Xie et al.** "Deep Learning for Image Colorization." 2017.