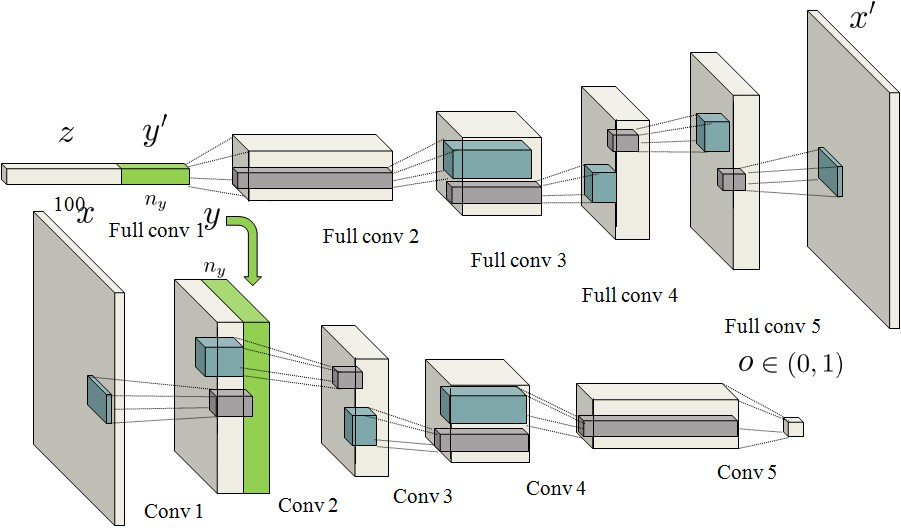


**A Generative Adversarial Networks**

**1. Introduction**

Generative Adversarial Networks (GANs) have revolutionized the field of generative modeling by enabling the synthesis of highly realistic images. A Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) extends the GAN framework by incorporating class labels into both the generator and discriminator, allowing for controlled image generation based on specified categories. This report details the implementation of a cDCGAN trained on the CIFAR-10 and MNIST datasets, highlighting data preparation, model architecture, training procedures, and evaluation metrics to assess performance.



Conditional DC GAN architecture

**2. Methodology:**

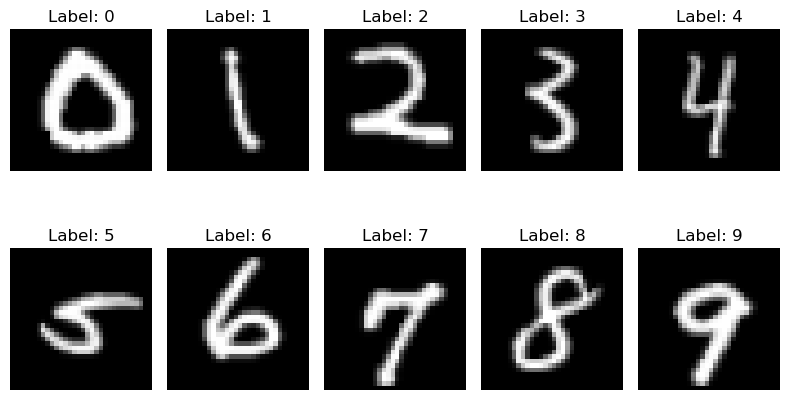
**Data Preprocessing**

**Datasets Used:**

1. **CIFAR-10:** Comprises 60,000 32x32 color images across 10 classes, including categories such as airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.
2. **MNIST:** Consists of 70,000 28x28 grayscale images of handwritten digits ranging from 0 to 9.

**Preprocessing Steps:**

* **Resizing:** MNIST images were resized to 32x32 pixels to align with the CIFAR-10 image dimensions, ensuring consistency in input size for the model.
* **Normalization:** Both datasets were normalized to the range [-1, 1] to facilitate stable training of the neural networks. This normalization was achieved by scaling the pixel values accordingly.
* **Data Loading:** The datasets were loaded separately using PyTorch's DataLoader with a batch size of 64 and shuffling enabled to ensure randomized input batches during training.



*Figure 1: MNIST Dataset*

**4. Architecture Design**

The cDCGAN comprises two primary components: the generator and the discriminator. Both networks are conditioned on class labels, enabling the generation of images corresponding to specific categories.

**Generator Architecture**

The generator network is responsible for producing realistic images conditioned on given class labels. Key components of the generator include:

* **Embedding Layer:** Transforms class labels into dense vectors that are concatenated with the latent noise vector. This conditioning allows the generator to produce images corresponding to specific classes.
* **Fully Connected Layer:** Combines the latent noise vector with the label embeddings, projecting them into a higher-dimensional space suitable for convolutional operations.
* **Convolutional Blocks:** Utilize transposed convolutions, batch normalization, and activation functions to upscale the feature maps progressively, ultimately generating images of the desired size and color channels.
* **Activation Function:** A Tanh activation function is applied at the output layer to ensure the generated images have pixel values within the normalized range of [-1, 1].

**Discriminator Architecture**

The discriminator network evaluates the authenticity of images while considering the provided class labels. Its role is to distinguish between real images from the dataset and fake images generated by the generator. Key components include:

* **Embedding Layer:** Transforms class labels into dense vectors that are reshaped and concatenated with the input images, enabling the discriminator to assess the relevance of the class label to the image.
* **Convolutional Blocks:** Employ standard convolutional layers, batch normalization, LeakyReLU activations, and dropout for regularization. These layers extract hierarchical features from the input images.
* **Output Layer:** Produces a single scalar value with a Sigmoid activation function, indicating the probability of the input image being real.

**Modified Architecture (with Dropout)**

The cDCGAN model's architecture incorporates dropout layers in the discriminator to prevent overfitting, while experimentation with learning rates involved reducing the discriminator's learning rate to ensure balanced adversarial training.

This adjustment aims to avoid a scenario where the discriminator quickly overpowers the generator, hindering the generator's ability to learn effectively. By applying a slightly lower learning rate to the discriminator, the generator and discriminator achieved a stable training balance, allowing both networks to improve in parallel.

**5. Training and Evaluation**

**Hyperparameters:**

* **Latent Vector Size (nz):** 100
* **Number of Channels (nc):** 1 for MNIST (grayscale), 3 for CIFAR-10 (color)
* **Generator Feature Maps (ngf):** 64
* **Discriminator Feature Maps (ndf):** 64
* **Number of Classes (num\_classes):** 10
* **Number of Epochs (num\_epochs):** 100
* **Learning Rate (lr):** 0.00002
* **Beta1 for Adam Optimizer:** 0.5

**Optimization:**

* **Loss Function:** Binary Cross Entropy Loss (BCELoss) was employed to measure the performance of both the generator and discriminator.
* **Optimizers:**
  + **Generator:** Utilized the Adam optimizer with a learning rate of lr and betas (beta1, 0.999).
  + **Discriminator:** Utilized the Adam optimizer with a learning rate of lr \* 0.1 and betas (beta1, 0.999). A lower learning rate for the discriminator was chosen to ensure balanced training between the generator and discriminator.

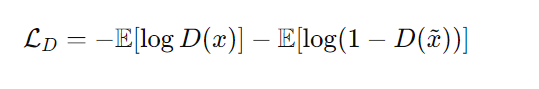
**Training Steps:**

1. **Generator Training:**
   * The generator creates fake images conditioned on randomly sampled class labels.
   * The discriminator evaluates these fake images, and the generator's loss is computed based on the discriminator's assessment.
   * The generator's weights are updated to minimize this loss, encouraging it to produce more realistic images.
   * The generator’s loss is based on the discriminator’s probability of classifying generated images as real. For a generated image with label y=1, the generator’s objective is to ‘fool’ the discriminator by maximizing D(x̅):



*Generator Loss*

1. **Discriminator Training:**
   * The discriminator evaluates real images from the dataset alongside their corresponding labels.
   * It also evaluates fake images generated by the generator with associated labels.
   * The discriminator's loss is computed as the average of its performance on real and fake images.
   * The discriminator's weights are updated to improve its ability to distinguish between real and fake images accurately.
   * Given a real image x with label y=1, and a generated (fake) image with label y=0, the loss is



*Discriminator Loss*

**Combined** **Loss function:**

****

*Combined Loss Function*

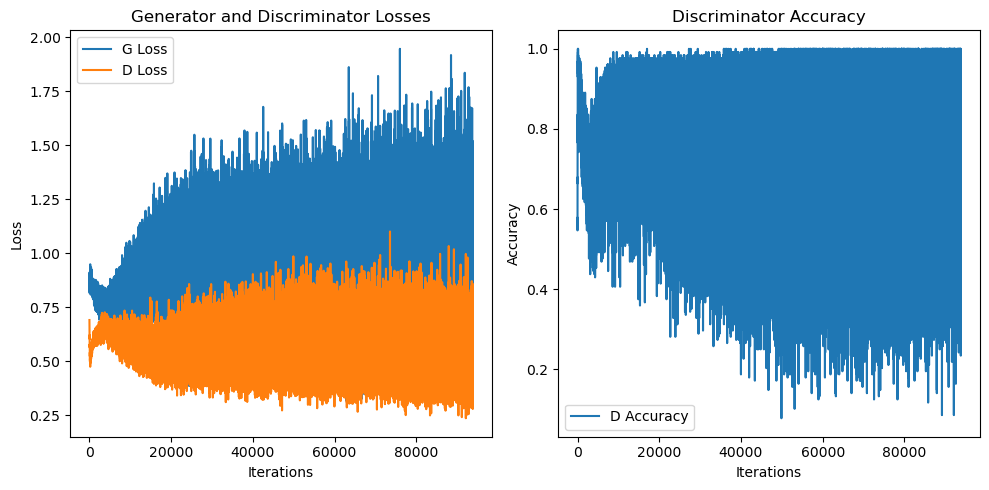
1. **Metrics Tracking:**
   * Both generator and discriminator losses were recorded throughout training.
   * Discriminator accuracy, representing its ability to correctly classify real and fake images, was also tracked.

**5.3 Results**

**Loss Curves**

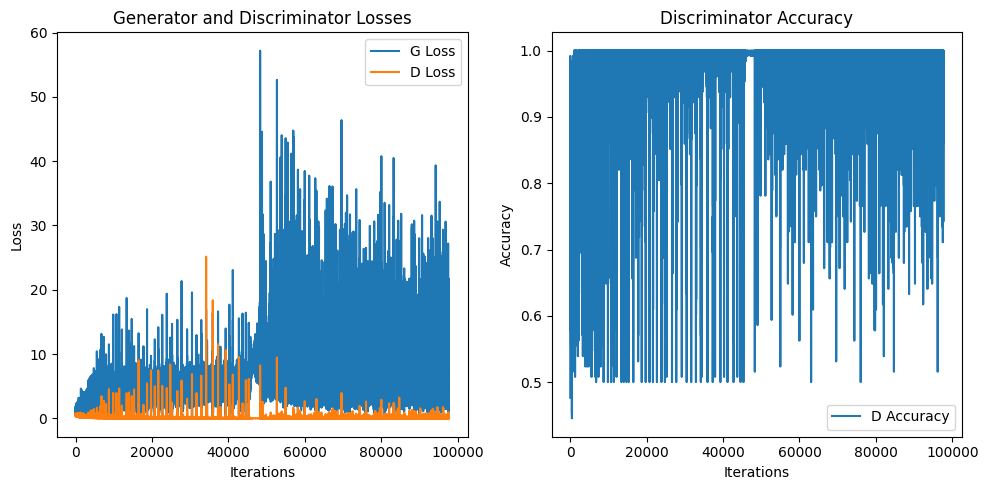
The generator and discriminator losses were plotted over the training iterations to evaluate convergence and training stability. Typically, the generator loss exhibited a decreasing trend, indicating improving generation quality, while the discriminator loss stabilized, reflecting a balanced adversarial interaction between the two networks.

(i) **For MINST:**



*Figure: Generator vs Discriminator loss and Discriminator Accuracy with iterations*

(ii) **For Cifar 10:**



*Figure: Generator vs Discriminator loss and Discriminator Accuracy with iterations*

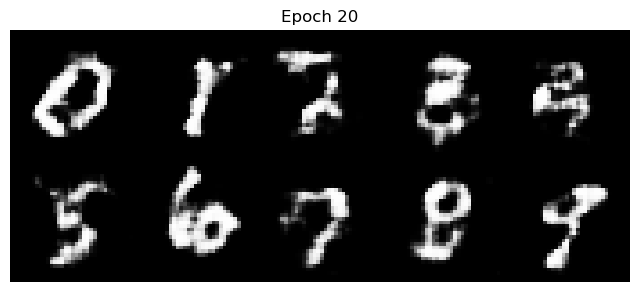
**Discriminator Accuracy**

Discriminator accuracy was tracked to ensure it maintained an effective balance in distinguishing real and fake images. Initially, high accuracy indicated the discriminator's proficiency, but as training progressed, accuracy fluctuations around 50% signified the generator's enhancement in producing realistic images, thereby challenging the discriminator.

**Generated Images**

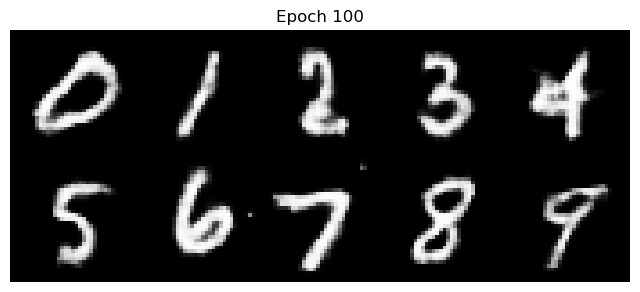
Generated images were qualitatively assessed at various training epochs to evaluate the cDCGAN's performance:

* **MNIST:**
  + The generator effectively produced clear and recognizable digits corresponding to the input labels.



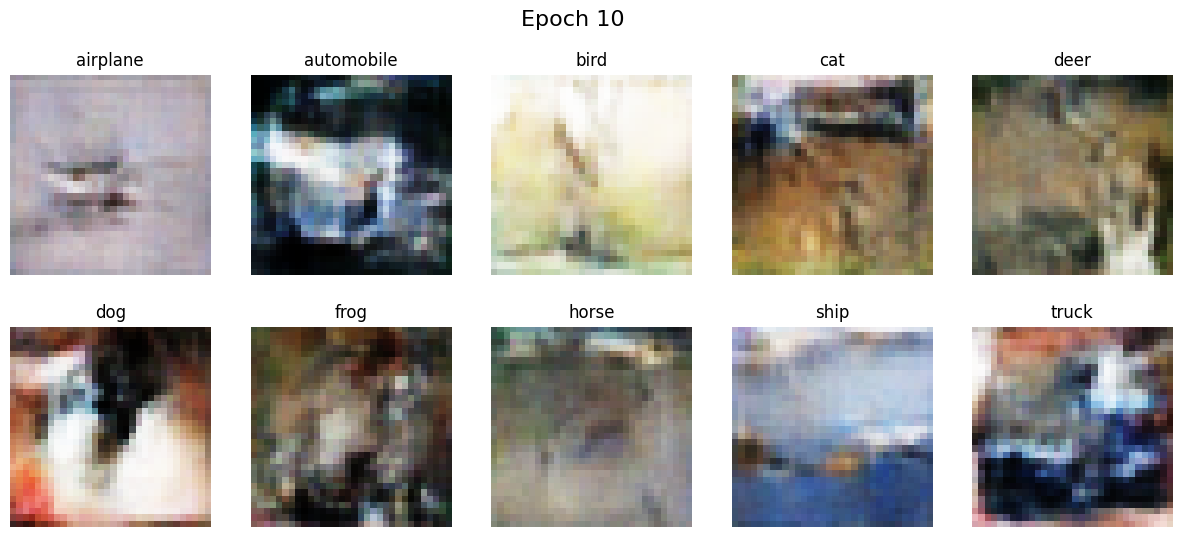
*Figure: Visualizing the outputs after 20 epochs.*

* + Over time, the images became sharper and more defined, closely resembling genuine handwritten digits.



*Figure: Visualizing the outputs after 100 epochs.*

* **CIFAR-10:**
  + The generator produced diverse and more complex images aligning with CIFAR-10 classes.



*Figure: Visualizing the outputs after 10 epochs.*

* + While many images captured the essence of their respective classes, some artifacts and inconsistencies remained, reflecting the dataset's higher complexity compared to MNIST.



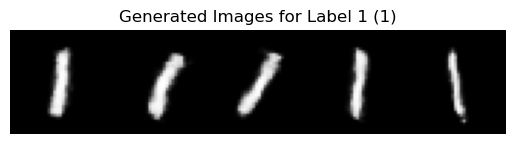
*Figure: Visualizing the outputs after 125 epochs.*

**Creating Class Specific Images:**

I also created a function which takes in the label and outputs images of the given label by passing it through the generator.

(i) **For MINST:**





*Figure: Visualizing some images of class labels from the generator.*

(ii) **For Cifar10:**

****

****

*Figure: Visualizing some images of class labels from the generator.*

**Evaluation Metrics**

To quantitatively assess the quality of generated images, two widely recognized metrics were employed: Inception Score (IS) and Frechet Inception Distance (FID).

**Inception Score**

Inception Score evaluates both the diversity and quality of generated images. Higher scores indicate better performance.

* **MNIST:**
  + **Inception Score:** 2.2988 ± 0.0310
* **CIFAR-10:**
  + **Inception Score:** 2.7618 ± 0.0309

**Interpretation:**

* **MNIST:** The Inception Score reflects the generator's ability to produce distinct and diverse digit images.
* **CIFAR-10:** A higher Inception Score suggests improved diversity and quality in generated images, albeit still facing challenges due to the dataset's complexity.

**Frechet Inception Distance (FID)**

FID measures the distance between the distributions of real and generated images, with lower scores indicating greater similarity and higher quality.

* **MNIST:**
  + **FID:** 46.9460
* **CIFAR-10:**
  + **FID:** 202.5178

**Interpretation:**

* **MNIST:** The relatively lower FID indicates that the generated digit images closely resemble the real distribution, highlighting effective training.
* **CIFAR-10:** The higher FID suggests significant differences between the generated and real image distributions, pointing to areas for improvement in capturing the dataset's variability.

**8. Conclusion**

This project successfully implemented a Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) capable of generating images conditioned on specific class labels for both MNIST and CIFAR-10 datasets. The model demonstrated proficiency in generating clear and recognizable digits for MNIST, as evidenced by satisfactory Inception Scores and FID values. However, generating complex and diverse images for CIFAR-10 presented greater challenges, indicated by higher FID scores. Future work may explore advanced architectural enhancements, improved training techniques, and larger-scale datasets to further enhance the quality and diversity of generated images. Additionally, incorporating techniques such as spectral normalization, progressive training, or attention mechanisms could potentially address the complexities observed in CIFAR-10 image generation.

*---------------------------------------------------- Report Ends --------------------------------------------------------*