WiDS - 2024 | Analytics Club

This Person Does Not Exist: An Intro to Generative Models

PRESENTED BY:
GURU JAHNAVI MADANA
ENGINEERING PHYSICS - 24B1809

MENTORED BY:
VEDANT BHARADWAJ
TUSHAR SINGHA ROY

Project UID: 100

Table of contents

INTRODUCTION

This project explores the fascinating world of generative models, focusing on Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). The primary motivation behind this study was to understand how artificial intelligence can generate realistic synthetic images and its applications in fields such as image synthesis, data augmentation, and anomaly detection.

A significant part of the project involved analyzing ThisPersonDoesNotExist.com, a website powered by StyleGAN, which produces hyper-realistic faces that do not belong to any real person. By delving into the architecture behind GANs and VAEs, implementing Deep Convolutional GANs (DCGANs) on datasets like MNIST and CIFAR-10, and experimenting with latent space representations, we gained deeper insights into how generative models function.

The study also tackled various challenges, such as mode collapse in GANs, loss function optimization, and computational constraints. The findings of this research serve as a foundation for further exploration into advanced generative models like diffusion models and hybrid VAE-GAN architectures. This report systematically documents our approach, methodologies, challenges, and key learnings.

WEEK -01

Understanding Generative Models

During the first week, we focused on understanding the fundamentals of Generative Models and their role in AI applications. Here's what we covered:

Background on Generative Models

Generative models are a class of machine learning models that learn to produce new data points that resemble an existing dataset. Unlike traditional discriminative models that classify or predict given an input, generative models learn the underlying distribution of the dataset and generate new samples from it.

Categories of Generative Models

1. *Explicit Density Models* – Directly estimate the probability distribution of the data. Examples:
    - Variational Autoencoders (VAEs) – Learn a compressed latent space representation of data.
    - Normalizing Flows – Transform a simple probability distribution into a complex one.
2. *Implicit Density Models* – Generate samples without explicitly defining a probability function. Examples:
    - Generative Adversarial Networks (GANs) – Use two neural networks in a competitive game to generate high-quality data.
    - Diffusion Models – Learn to generate images by reversing a noise corruption process.

Deep Learning and Neural Networks

- Deep learning is a subset of machine learning that utilizes artificial neural networks to model complex patterns and representations.
- A neural network is composed of multiple layers of interconnected neurons that process and transform input data.
- The fundamental building blocks include:
  - Input layer: Receives raw data.
  - Hidden layers: Perform transformations using weighted connections and activation functions.
  - Output layer: Produces final predictions or classifications.

Convolutional Neural Networks (CNNs)

- CNNs are a specialized type of neural network primarily used for image processing tasks.
- They consist of convolutional layers that apply filters to detect patterns, pooling layers for dimensionality reduction, and fully connected layers for classification.
- CNNs are highly effective in feature extraction and have revolutionized computer vision applications.

Introduction to AI-Generated Content:

- Explored ThisPersonDoesNotExist.com and how it generates fake human faces.
- Understood the underlying principles of StyleGAN and how it produces high-quality synthetic images.
- Studied other "X Does Not Exist" websites that generate non-existent objects, cats, and even rental listings.

Basic Concepts of Generative Models

- Difference between generative and discriminative models.
- Importance of learning data distributions rather than direct classification.
- Real-world applications like deepfake technology, AI-driven art, and medical data synthesis.

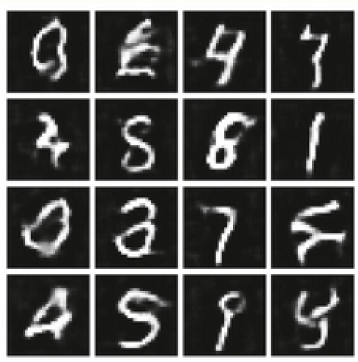
Introduction to PyTorch and Its Role in Generative Models:

- Studied PyTorch, a deep learning framework, and its advantages over other frameworks like TensorFlow.
- Learned about Tensors in PyTorch, how they serve as the building blocks for deep learning models.
- Implemented basic GAN architectures in PyTorch, understanding its dynamic computation graph feature.
- Explored optimizers like Adam and RMSprop in PyTorch for training neural networks.
- Used PyTorch's autograd feature for automatic differentiation and gradient tracking.
- Understood how PyTorch's GPU acceleration allows for faster training of generative models.

Assignment: Implementing DCGAN on MNIST

- Implemented a Deep Convolutional GAN (DCGAN) on the MNIST dataset to generate handwritten digits.
- Used convolutional layers and batch normalization to improve stability and training efficiency.
- Trained the model using Adam optimizer with a learning rate of 0.0002 and experimented with different batch sizes.
- Observed the loss curves for both the generator and discriminator and analyzed how changing hyperparameters affected the image quality.
- Final results showed clear and recognizable digit generation, demonstrating the ability of DCGANs to learn complex distributions.

Assignment link - https://github.com/GJ-007-sage/Generative-Models-/blob/main/DCGANs_MNIST.ipynb


WEEK -02

Deep Dive into Generative Adversarial Networks (GANs)

In the second week, we explored Generative Adversarial Networks (GANs) in depth, understanding their structure, working principles, and significance in generative modeling.

Understanding Generative Adversarial Networks (GANs)

GANs are a class of generative models that use a two-network system—Generator and Discriminator—to generate realistic samples from noise. They were introduced by Ian Goodfellow in 2014 and have since become a cornerstone of generative AI.

Structure of GANs

GANs consist of two main components:

- Generator (G): Takes random noise as input and generates synthetic data resembling the training dataset.
- Discriminator (D): Acts as a classifier that distinguishes between real and generated data.

These two networks compete in a zero-sum game where:

- The Generator tries to fool the Discriminator by producing increasingly realistic samples.
- The Discriminator improves by correctly identifying fake and real samples.

Mathematical Formulation of GANs

Understanding Probability Distributions:
Generative models learn an unknown data distribution $p_{data}(x)$ by estimating a function $p_{model}(x)$, so that $p_{model}(x)$ is as close as possible to the real data distribution.

In GANs, this is achieved through a min-max game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

where:

- G(z) generates fake data from noise .
- D(x) predicts if an image is real or fake.
- $p_{data}(x)$ represents the real data distribution.
- $p_z(z)$ is the noise distribution from which the generator samples.

Training GANs: The Adversarial Process

The training process involves the following steps:

1. Sample real data from the dataset.
2. Generate fake data from the Generator using random noise.
3. Train the Discriminator to correctly classify real vs. fake data.
4. Train the Generator to improve its ability to fool the Discriminator.
5. Repeat the above steps until convergence, ideally when the Discriminator can no longer distinguish between real and fake samples.

Challenges in Training GANs

- Mode Collapse: The Generator produces limited variations instead of diverse samples.
- Training Instability: GANs require careful tuning of hyperparameters, loss functions, and network architectures.
- Vanishing Gradient Problem: If the Discriminator becomes too strong, it stops providing meaningful gradients to the Generator.

Improvements in GANs

To address these challenges, researchers have developed improved versions of GANs:

- DCGAN (Deep Convolutional GANs): Uses convolutional layers instead of fully connected layers for better feature extraction.
- WGAN (Wasserstein GAN): Introduces Wasserstein distance for stable training.
- StyleGAN: Generates high-quality human faces using hierarchical latent space transformations.

GANs have numerous applications, including image synthesis, super-resolution, data augmentation, and video generation.

Assignment

- **DCGAN on CIFAR-10**: Implemented a DCGAN model trained on the CIFAR-10 dataset to generate realistic images. This involved experimenting with convolutional layers and hyperparameters to improve stability and output quality.

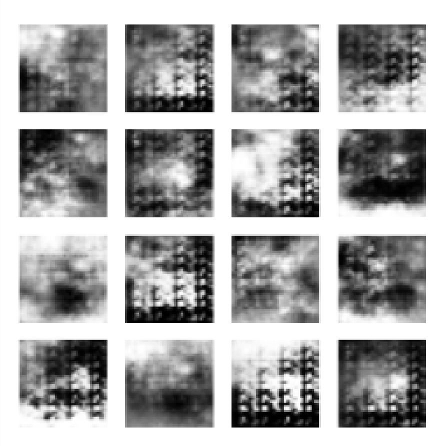
https://github.com/GJ-007-sage/Generative-Models-/blob/main/CIFAR10_RGB.ipynb

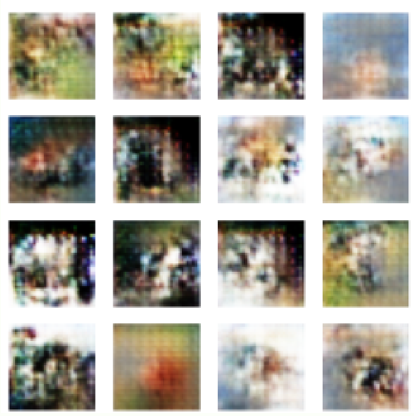
- **Advanced Assignment on GANs**: Conducted additional experiments to improve GAN performance by fine-tuning network architectures, batch sizes, and optimizers, aiming for more stable and higher-quality image generation.

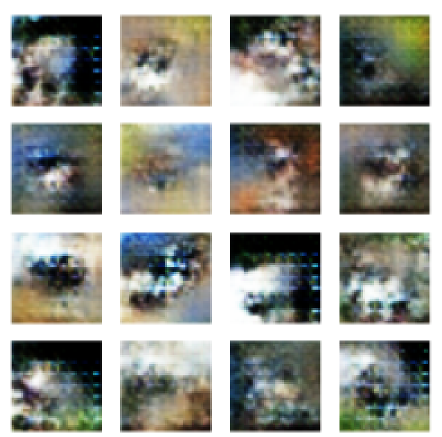
https://github.com/GJ-007-sage/Generative-Models-/blob/main/AAssignment_02.ipynb

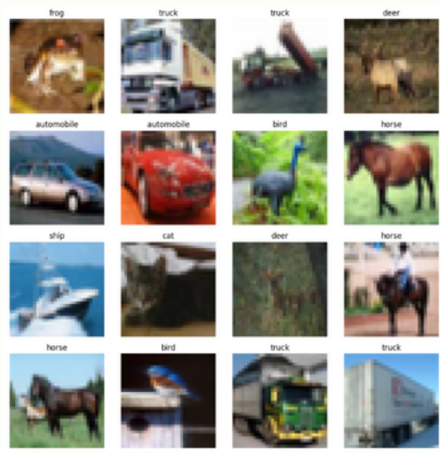
- **Assignment on Loss Functions:** Analyzed different loss functions used in GANs, including Binary Cross-Entropy and Wasserstein loss, to understand their impact on model convergence and training efficiency.

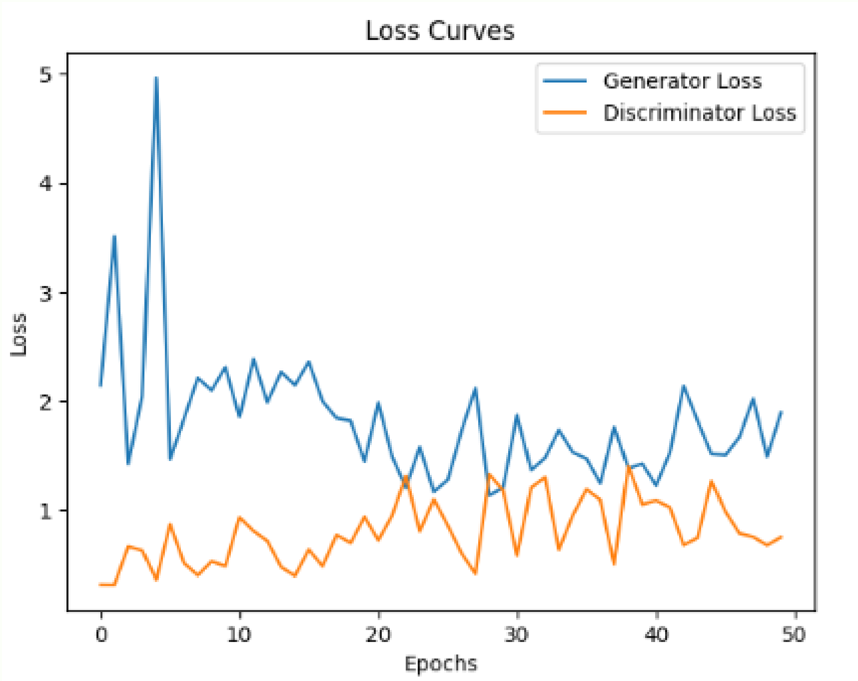
https://github.com/GJ-007-sage/Generative-Models-/blob/main/Assignment.ipynb








frog
truck
truck
deer
automobile
automobile
bird
horse
ship
cat
deer
horse
horse
bird
truck
truck


Loss Curves
Generator Loss
Discriminator Loss
Loss
Epochs
0
10
20
30
40
50
1
2
3
4
5

WEEK -03

Variational Autoencoders (VAEs)

In the third week, we focused on Variational Autoencoders (VAEs), another important class of generative models that rely on probabilistic frameworks to encode and generate data.

Understanding Variational Autoencoders (VAEs)

VAEs are a type of autoencoder that introduces a probabilistic element to the latent space, allowing them to generate diverse and continuous variations of data samples.

Structure of VAEs

- Encoder: Compresses input data into a latent space representation.
- Latent Space Sampling: Instead of mapping inputs to fixed points, VAEs model them as probability distributions (usually Gaussian), introducing a stochastic element.
- Decoder: Reconstructs data samples from latent space representations, ensuring generated samples resemble real data.

Mathematical Formulation of VAEs

VAEs optimize a loss function consisting of two terms:

$$\mathcal{L}(x, z) = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}(q(z|x)||p(z))$$

where:

- The first term represents reconstruction loss, ensuring the generated output is close to the original input.
- The second term is KL divergence, which regularizes the latent space by ensuring it follows a known prior distribution (typically Gaussian).

Latent Space Representation

- Unlike standard autoencoders, which learn deterministic embeddings, VAEs introduce stochasticity, ensuring smoother and more meaningful latent representations.
- This enables controlled interpolations between samples, making VAEs highly effective in data synthesis.

Applications of VAEs

- Image Generation: Creating new realistic images by sampling from learned distributions.
- Anomaly Detection: Identifying unusual patterns by reconstructing input data and measuring deviations.
- Data Compression: Efficiently encoding data in a reduced-dimensional latent space.
- Text & Speech Synthesis: Generating human-like text and speech from structured latent spaces.

Challenges in VAEs

- Blurry Outputs: Unlike GANs, VAEs tend to produce smoother but sometimes less detailed images.
- Balancing KL Divergence: Poor regularization can lead to either overly compressed or underutilized latent spaces.

Key Takeaways from Week 3

- VAEs provide a structured approach to generative modeling, incorporating probabilistic elements for better representation learning.
- The balance between reconstruction loss and KL divergence is crucial for achieving meaningful latent spaces.
- VAEs complement GANs by offering interpretable latent representations, making them useful for structured data generation.

Sources and References

- https://thispersondoesnotexist.com/
- https://azure-fragrance-241.notion.site/DGMs-An-Odyssey-329e06a821b34c549f55c040c8e8cfde
- https://github.com/soumith/ganhacks
- https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- https://instinctive-pyjama-015.notion.site/WiDS-Plan-1359ce235408808e9ccdd43ccf6df8a7
- https://instinctive-pyjama-015.notion.site/Generative-Models-1639ce23540880bb89e1c9ed9defe460
- https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

Git hub repo

https://github.com/GJ-007-sage/Generative-Models-/tree/main

Conclusion

This project provided a comprehensive understanding of Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), their architectures, and their applications in artificial intelligence. GANs proved effective in generating high-resolution, realistic images, while VAEs offered a structured approach to latent space representation and controlled data synthesis. Implementing these models helped me grasp key concepts such as adversarial training, loss function optimization, and challenges like mode collapse and instability in GANs.

Hands-on experience with PyTorch allowed me to experiment with different architectures, fine-tune hyperparameters, and analyze the impact of various loss functions on model performance. Additionally, exploring ThisPersonDoesNotExist.com deepened my understanding of how StyleGAN leverages progressive training and instance normalization to generate high-quality synthetic faces.

Beyond the technical aspects, this project also emphasized the ethical implications of AI-generated content, particularly in areas like deepfakes and misinformation. Moving forward, I aim to explore Diffusion Models, hybrid VAE-GAN architectures, and improved interpretability techniques to push the boundaries of generative modeling and its practical applications.

> “GANs are a fascinating approach to generative modeling, where two neural networks compete to create something entirely new.”
>
> -IAN GOODFELLOW, INVENTOR OF GANS