Report: Deep Learning Lab7

Ying LAI, Yingjie LIU

April 2024

1 Introduction

This report presents our findings from experimenting with Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN) on the MNIST dataset using PyTorch. We explored these generative models to synthesize and enhance images of handwritten digits. Our VAE implementation focuses on optimizing the latent space dimensions and reconstruction loss, while our GAN experiments involve refining generator and discriminator architectures for improved image quality. We evaluate both models based on their effectiveness in producing clear and accurate digit images, highlighting the key configurations and their impacts on the output.

2 Network Architecture

2.1 Variational Auto-Encoder (VAE)

The Variational Auto-Encoder (VAE) is a cornerstone of modern generative models, effectively balancing the capabilities of deep learning with the principles of statistical inference. At its core, the VAE is designed to learn the probability distribution of a dataset by encoding data into a latent space and then reconstructing the data from this space.

The VAE framework hinges on the assumption that the data are generated by a random process involving an unobserved continuous random variable Z. The data generation process can be decomposed as:

1. **Latent Variable Model:**
$$Z \sim p_\theta(z)$$
where Z is the latent variable and $p_\theta(z)$ s the prior distribution, typically chosen to be a standard Gaussian $\mathcal{N}(0, I)$

2. **Conditional Likelihood:**
$$X \sim p_\theta(x|z)$$
where X is the observed data generated from the latent variable Z. The conditional likelihood $p_\theta(x|z)$ s modeled by a decoder network, which maps the latent variable back to the data space.

Then to enable efficient gradient-based optimization, VAEs use the reparameterization trick, which redefines the random variable Z in a way that allows gradients to pass through deterministic nodes:

$$\epsilon \sim \mathcal{N}(0, I), \quad z = \mu + \sigma \circ \epsilon$$

Here, ϵ is an auxiliary noise variable, μ and σ are outputs from the encoder network that predict the mean and standard deviation of the latent variables' distribution.

The optimization of a Variational Auto-Encoder is guided by two main loss functions, which collectively ensure that the encoder and decoder networks effectively learn the data distribution:

1. **Reconstruction Loss L_{recon}:** The reconstruction loss quantifies the error in reconstructing the original input data from the compressed latent representation. It is typically calculated using the Mean Squared Error (MSE) between the original images (x) and the reconstructed images ($\hat{x}$), aiming to minimize the differences between them:
$$L_{recon} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{x}_i)^2.$$

2. **KL Divergence L_{KL}:** The KL Divergence acts as a regularization term and measures the divergence between the learned latent variable distribution $q(z|x)$ and the prior distribution $p(z)$. This divergence helps ensure that the encoder distributes the latent variables in a manner that closely approximates the assumed prior (usually a standard Gaussian distribution). The KL Divergence for each element in the batch is given by:

$$L_{KL} = -\frac{1}{2}\sum(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

Combining these two components, the total VAE loss function, which the training process seeks to minimize, is:

$$L_{VAE} = L_{recon} + L_{KL}.$$

This loss formulation ensures that the VAE not only faithfully reconstructs the input data but also effectively regularizes the distribution of the latent variables, thereby enabling the model to generate new data points that are consistent with the input data distribution.

2.2 GAN (Brief Overview)

In contrast, Generative Adversarial Networks (GANs) involve training two models: a generator that creates images from noise, and a discriminator that attempts to distinguish between real and generated images. GANs do not involve explicit encoding and decoding processes or a latent space structured similarly to VAEs. Instead, they focus on adversarial training to produce high-quality generative models, which can sometimes result in more visually appealing outputs than those of VAEs.

3 Implementation

The VAE model in our implementation consists of two main components: the encoder and the decoder, which are structured to compress and then reconstruct the input data, optimizing both the latent space representation and the reconstruction accuracy.

The encoder is designed to map input data into a latent space characterized by two parameters: mean (μ) and log variance ($log(\sigma)$). It comprises sequential fully connected layers followed by ReLU activations, with the final layer splitting into two pathways to output both parameters.

```
class VAEEncoder(nn.Module):
    def __init__(self, dim_input, dim_latent):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(dim_input, 256),  # First layer
            nn.ReLU(),                  # Activation
            nn.Linear(256, dim_latent * 2),  # Outputs mean and log-variance
        )

    def forward(self, inputs):
        code = inputs.view(inputs.size(0), -1)
        code = self.encoder(code)
        mu, log_sigma_squared = code.chunk(2, dim=1)  # Split output into mean and log
    variance
        return mu, log_sigma_squared
```

The decoder takes the latent variables and reconstructs the input data, aiming to minimize the reconstruction loss. It inversely mirrors the encoder's architecture.

```
    class VAEDecoder(nn.Module):
    def __init__(self, dim_latent, dim_output):
        super().__init__()
        self.decoder = nn.Sequential(
            nn.Linear(dim_latent, 256),
            nn.ReLU(),
            nn.Linear(256, np.prod(dim_output)),  # Matches output to input dimensions
            nn.Sigmoid()  # Normalizes output to pixel range
        )

    def forward(self, z):
        output = self.decoder(z)
        img = output.view(-1, 1, 28, 28)  # Reshape output to image dimensions
        return img
```

For the training process, the total loss for the VAE is the sum of the reconstruction loss and the KL divergence. The reconstruction loss is computed as the Mean Squared Error between the original and reconstructed images, while the KL divergence ensures the distribution of the latent variables aligns with the assumed Gaussian prior. The network utilizes the Adam optimizer with a learning rate typically set at 0.001. Training involves backpropagation where gradients are computed for both loss components and used to update the network weights.

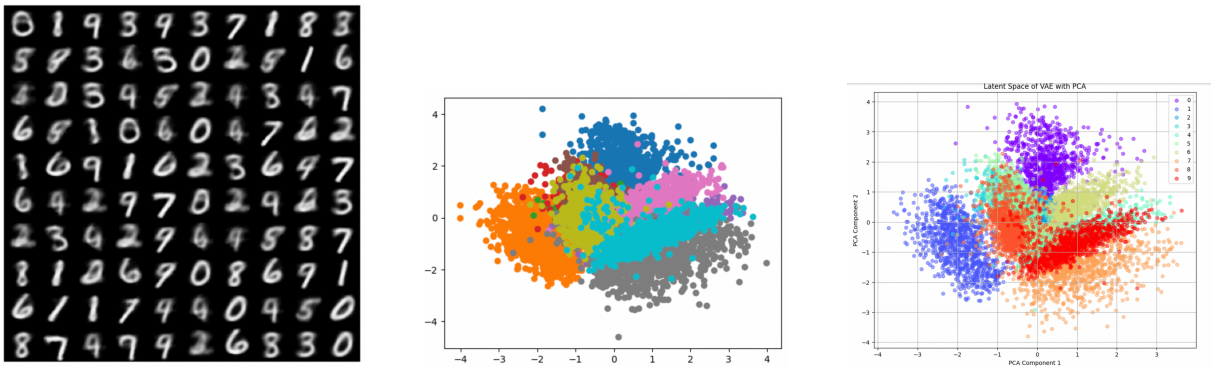
In contrast, the GAN setup involves a generator creating images from noise and a discriminator assessing their authenticity against real dataset images. While GANs are part of the study, the primary focus remains on refining and understanding the VAE due to its implications for understanding deep generative models.

4 Experiments and Results

4.1 VAE Experiments and Results

The VAE experiments included a detailed analysis of the latent space, with PCA visualizations showcasing how different digit representations clustered distinctly. The figure(1b) shows latent space and figure(1c) is latent space of PCA. This indicates a well-defined latent space where similar digits are mapped close together, demonstrating the model's effective learning and differentiation capabilities. Latent space visualization proved particularly insightful, as it highlighted how the VAE managed to organize the latent representations of the MNIST digits in a manner that preserves their categorical identity despite the compression.

Reconstruction accuracy improved progressively as the VAE was trained. Initial outputs were blurry and less distinct, but as the epochs advanced, the clarity and accuracy of the reconstructed images significantly improved, suggesting an effective tuning of the network parameters. The figure(1a) shows the final image we generated. Throughout the training, the VAE loss consistently declined, reflecting steady enhancements in both reconstruction quality and adherence to the latent space's prior distribution.



(a) Picture generated by VAE (b) Latent Space (c) Latent Space of PCA

Figure 1: The results of VAE

4.2 Comparison of GAN and VAE Results

Turning to the GAN, its initial performance starkly contrasted with that of the VAE. The early generated images by the GAN were primarily noise, lacking discernible features2. However, iterative refinements to the network architecture and training parameters gradually led to improvements. Unlike the VAE's stable decrease in loss, the GAN's training was characterized by notable fluctuations in both the discriminator and generator losses.

For instance, while discriminator loss occasionally showed a decreasing trend, suggesting improved real versus fake classification, generator loss varied unpredictably, peaking significantly at certain epochs before dipping. This pattern suggests ongoing challenges in stabilizing the GAN's training dynamics. Such fluctuations underscore the difficulties in training GANs to achieve a balance where the generator produces realistic outputs while the discriminator remains effectively critical.

Despite improvements, the final images3 generated by the GAN, although clearer than the initial outputs, still did not reach the level of detail and consistency achieved by the VAE. The GAN strug-

gled with maintaining consistent quality across training epochs, with the generator loss often spiking dramatically, indicating that the generator fluctuated between underfitting and momentarily generating plausible outputs before regressing again.

These experiments underscore the relative robustness and reliability of VAEs in generating and reconstructing images as compared to GANs, which, while powerful, require careful tuning and are more sensitive to training instability. This comparative analysis not only highlights the strengths and weaknesses of each model but also underscores the practical challenges faced when deploying these advanced generative techniques in real-world applications.

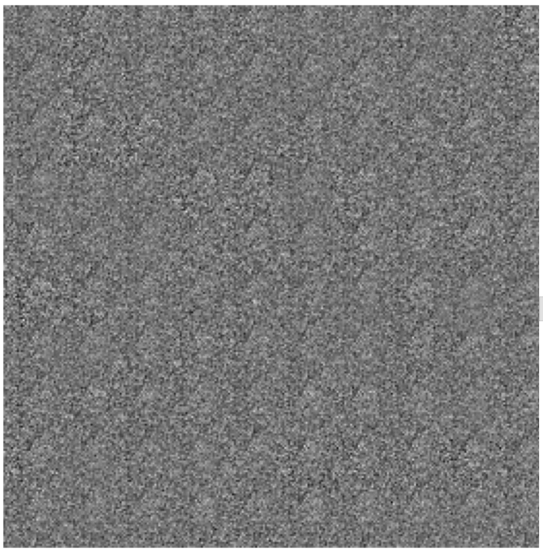


Figure 2: Bad result generated by GAN

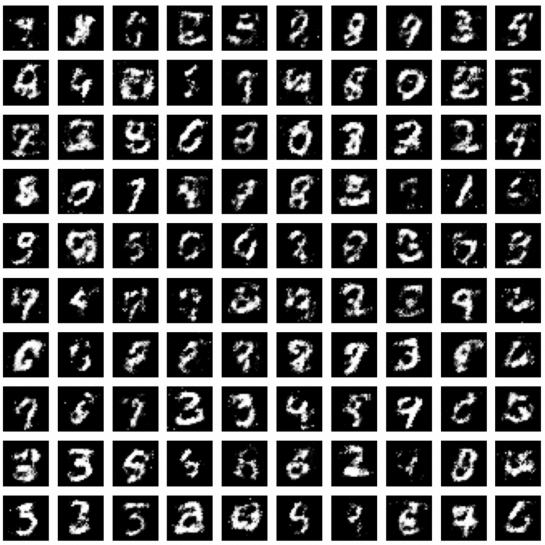


Figure 3: Result generated by modified GAN

5 Conclusion

Throughout our experiments with the Variational Auto-Encoder (VAE) on the MNIST dataset, the VAE demonstrated exceptional capability in encoding and reconstructing digital images. The clear clustering of different digit classes within the latent space, as visualized through PCA, confirmed the model's effectiveness in capturing distinct, intrinsic characteristics of the dataset. Additionally, the VAE's consistent performance improvement, as observed through progressively better image reconstructions and decreasing loss values, highlighted its robustness and reliability.

In comparison, the Generative Adversarial Network (GAN) showcased potential in image generation but faced challenges with stability and consistency. Despite adjustments aimed at stabilizing training, the GAN's results remained variable and generally did not match the fidelity achieved by the VAE.

6 Future Work

Future efforts will focus primarily on further enhancing the VAE's architecture and exploring additional applications:

Investigating more efficient network architectures or training procedures that could accelerate the learning process without sacrificing output quality. Application to More Complex Datasets: Applying the VAE to more complex or larger datasets will test the model's scalability and robustness, providing insights into potential areas for improvement. In addition, we could explore hybrid models that integrate the strengths of VAEs with other generative models like GANs could potentially yield improvements in image quality and diversity.