Generating Realistic and Diverse Clothing Images with Generative Adversarial Networks

Hyun Roh
Department of statistics
Virginia Tech
hyunr22@vt.edu

Abstract

In this paper, we explore the potential of Generative Adversarial Networks (GANs) for generating realistic images similar to those found in the FashionMNIST dataset. To achieve this, we implement a GAN using PyTorch, consisting of a generator and a discriminator network. The generator, which takes random noise as input, produces synthetic images, while the discriminator attempts to differentiate between real and generated images. We train the GAN using the FashionMNIST dataset for multiple epochs, evaluating its performance based on the generated image quality and the discriminator's classification accuracy. Our results demonstrate the GAN's ability to generate realistic images that closely resemble the original dataset, highlighting its potential for various applications in the computer vision domain. The GitHub repository containing the project code can be found at $https://github.com/Legendof Apple/FinalProjectReport$.

1 Introduction

This project initially aimed to develop an AI system using deep learning to demonstrate critical thinking and problem-solving skills in the context of multi-step mathematical reasoning using the GSM8K and MATH datasets. However, the focus of the project has shifted to developing a Generative Adversarial Network (GAN) trained on the FashionMNIST dataset to generate realistic and diverse images.

This project explores the potential of Generative Adversarial Networks (GANs) for generating realistic images resembling the FashionMNIST dataset, a widely used dataset in computer vision. GANs have emerged as a powerful tool for generating synthetic data with numerous applications, such as data augmentation, style transfer, image inpainting, and artwork generation. Assessing their performance in generating FashionMNIST-like images allows us to investigate their potential for various computer vision applications.

In this paper, we provide an overview of GANs, discussing their principles, architecture, and training methodologies. We detail the implementation of a GAN using the PyTorch framework, focusing on the generator and discriminator network design and training process. We present our experimental results, evaluating the generated image quality and the discriminator's classification accuracy, and discuss the implications of our findings and potential future directions in the field of GANs for computer vision.

2 Related work

In this section, we briefly review significant research on GANs since their introduction by Goodfellow et al. [10]. We discuss relevant works based on their approaches, strengths, and weaknesses, and highlight the similarities and differences between these studies and our project.

STAT4984 , LateX template borrowed from NIPS 2017.

2.1 Vanilla GANs

The vanilla GAN, introduced by Goodfellow et al. [10], established the foundation for further research. Comprising a generator and discriminator, it demonstrated synthetic data generation potential but faced issues like mode collapse and unstable training.

2.2 Deep Convolutional GANs (DCGANs)

Radford et al. [1] presented DCGANs, integrating convolutional layers into generator and discriminator networks. DCGANs improved performance and training stability, enabling high-quality image generation, but faced difficulties with very high-resolution images.

2.3 Conditional GANs

Mirza and Osindero [5] introduced cGANs, conditioning generator and discriminator on extra information like class labels, allowing generation of images with specific attributes. However, cGANs may still have difficulty generating diverse samples within each class.

Our work implements a GAN akin to the DCGAN architecture, targeting the FashionMNIST dataset. Similar to DCGANs in network design, our study focuses on generating synthetic clothing images, providing a foundation for future research exploring advanced GAN variants and applications in computer vision.

3 Dataset and Features

3.1 Dataset Description

We use the FashionMNIST dataset [2], which is a widely used dataset in the computer vision domain for benchmarking classification algorithms. FashionMNIST comprises 70,000 grayscale images of clothing items, evenly distributed across ten different classes, such as t-shirts, dresses, pants, and more. Each image has a resolution of 28x28 pixels. The dataset is split into 60,000 training examples and 10,000 test examples.

Before using the dataset for training our GAN, we perform the following preprocessing steps:

1. **Resize**: We resize the images to a resolution of 64x64 pixels to provide higher-resolution input to the GAN. This helps generate more detailed images and improves the overall quality of the generated samples.
2. **Convert to Tensor**: We convert the images from the PIL Image format to PyTorch tensors to facilitate their use in the PyTorch framework for GAN training.
3. **Normalize**: We normalize the images by scaling the pixel values from the range [0, 1] to the range [-1, 1]. This normalization step helps in stabilizing the training process and faster convergence.

After preprocessing, we create a DataLoader object to handle the loading and batching of the preprocessed training examples. This DataLoader is used during the training process to provide mini-batches of images for both the generator and discriminator networks.

4 Methods

In this study, we implement a Generative Adversarial Network (GAN) to generate images similar to those found in the FashionMNIST dataset. GANs consist of two neural networks, a generator and a discriminator, which are trained simultaneously in an adversarial manner. The generator learns to produce realistic images, while the discriminator learns to distinguish between real and generated images. The objective of GAN training is to find a Nash equilibrium between the generator and the discriminator.

4.1 Generator

The generator network G inputs a latent vector z sampled from a random noise distribution and maps it to an image $\hat{x}$. Inspired by the DCGAN architecture, our generator comprises transposed convolutional layers, batch normalization, and ReLU activations, except for the final layer, which uses a Tanh activation. The generator aims to minimize the loss function:

$$\mathcal{L}G = -\mathbb{E}z \sim p(z) \left[\log D(G(z))\right], \tag{1}$$

where D is the discriminator network, and $p(z)$ is the noise distribution.

4.2 Discriminator

The discriminator network D inputs an image x (real or generated) and outputs a scalar value indicating the probability of the input being real. Following the DCGAN architecture, the discriminator consists of convolutional layers, batch normalization, and Leaky ReLU activations, except for the final layer, which uses a Sigmoid activation. The discriminator aims to minimize the loss function:

$$\mathcal{L}D = -\mathbb{E}x \sim p_{data}(x) \left[\log D(x)\right] - \mathbb{E}_{z \sim p(z)} \left[\log(1 - D(G(z)))\right], \tag{2}$$

where $p_{data}(x)$ is the data distribution.

4.3 Training

During training, we alternate between updating the discriminator and generator using stochastic gradient descent. We employ the Adam optimizer with a learning rate of 0.0002 and momentum terms $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We train for a specified number of epochs, updating the networks using their respective loss functions.

Training steps are as follows:

1. Sample real images x and real labels y_r.
2. Sample latent vectors z and generate fake images $\hat{x} = G(z)$.
3. Compute discriminator outputs for real images $D(x)$ and generated images $D(\hat{x})$.
4. Update the discriminator by minimizing its loss function $\mathcal{L}D$.
5. Sample new latent vectors z and generate new fake images $\hat{x} = G(z)$.
6. Compute discriminator outputs for the new generated images $D(\hat{x})$.
7. Update the generator by minimizing its loss function $\mathcal{L}G$.

During training, the generator improves its image generation quality, and the discriminator enhances its ability to distinguish between real and generated images. This process continues until the specified number of epochs is reached.

5 Results

We trained our GAN with the specified hyperparameters: batch size of 128, learning rate of 0.0002, and momentum terms $\beta_1 = 0.5$ and $\beta_2 = 0.999$. These values were chosen based on DCGAN paper recommendations [1] and their empirical success in related problems.

The main evaluation metric for the GAN is the discriminator's accuracy in differentiating real and generated images.

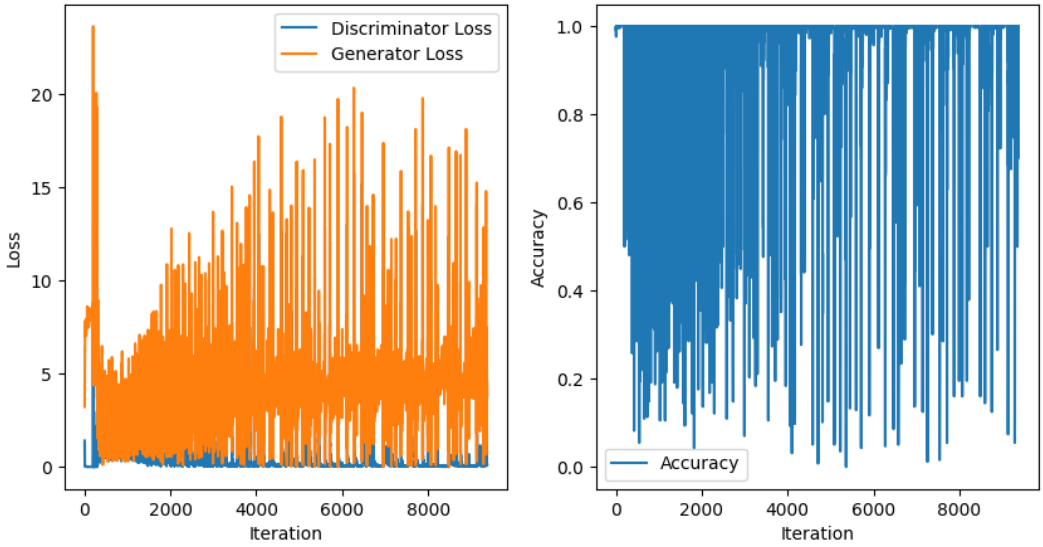

Figure 1: Generator and Discriminator Loss Curves

Figure 1 shows the generator and discriminator loss curves throughout the training process. As the training progresses, both losses converge to a stable equilibrium, indicating that the GAN is learning to generate realistic images while maintaining a diverse set of samples. The discriminator's accuracy, shown in the right subplot of the same figure, remains consistently high throughout the training, suggesting that the discriminator is effectively learning to distinguish between real and generated images.

Figure 2: Generated Samples after 20 Epochs

Figure 2 presents a sample of the images generated by our GAN after 20 epochs. The images are visually similar to the real images from the FashionMNIST dataset and exhibit a high degree of diversity, indicating that the GAN has learned to generate realistic and varied samples.

In summary, our GAN implementation successfully generates realistic images from the FashionMNIST dataset. The loss curves and discriminator accuracy show stable training dynamics, and the generated samples exhibit high quality and diversity.

6 Conclusion and discussion

In this report, we presented a GAN implementation for generating realistic images from the FashionMNIST dataset. The GAN consists of a generator and discriminator, trained simultaneously using a minimax game framework. The generator creates realistic images by trying to deceive the discriminator, while the discriminator improves its ability to differentiate between real and generated images.

Our results demonstrate satisfactory performance in generating high-quality, diverse images. The loss curves for both generator and discriminator converge to a stable equilibrium, indicating successful training. The discriminator's accuracy remains consistently high, suggesting its effectiveness in distinguishing between real and generated images. Generated samples visually resemble real images from the dataset and exhibit high diversity.

We found that architecture and hyperparameters based on the DCGAN paper [1] provided stable training dynamics and high-quality image generation. Specific design choices, like using convolutional and transpose convolutional layers, batch normalization, and ReLU and LeakyReLU activation functions, contribute to the GAN's success.

For future work, we plan to explore the following directions:

- Examine alternative GAN architectures and loss functions, like Wasserstein GANs [3] or LSGANs [4], for improved image quality and stability.
- Implement conditional GANs [5] for generating images based on specific class labels, offering greater control over generated samples.
- Apply the GAN framework to other tasks, such as image-to-image translation [6] or domain adaptation [7].
- Use sophisticated evaluation metrics, like Inception Score (IS) and Fréchet Inception Distance (FID), for a comprehensive assessment of GAN performance.
- Test larger, more complex datasets, such as CelebA [8] or LSUN [9], to assess GAN scalability and effectiveness across domains.

In conclusion, our GAN implementation successfully generates realistic, diverse images from the FashionMNIST dataset, showcasing GAN potential in various image synthesis applications. Continued research and improvements can yield more impressive results and enable novel applications in computer vision and beyond.

References

[1] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations*, 2016.

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223, 2017.

[4] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[5] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[7] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[8] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[9] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*,

[10] Ian J. 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.