Reproduce LayoutGAN: A Generating Graphic Layouts With Wireframe Discriminators

Aman Ayan Shah(as16600)[1], Jay Rana(jpr8961)[1], Meet Nirav Diwan(md5517)[1]
[1]New York University

GitHub Repository:
https://github.com/Meet1903/ECE-GY-6953-Final-Project/tree/main

Abstract

This project involved reproducing a LayoutGAN, originally proposed by Li et al. in 2019, architecture for generating layouts using a Generative Adversarial Network. LayoutGAN is a novel generative adversarial network (GAN) that automatically generates realistic and visually appealing layouts, improving the field of graphic design and scene generation. This GAN was proposed by Li et al.in 2019. There are updated versions of LayoutGAN also published afterward. We are reproducing the architecture proposed by Li et al. with modified Hyperparameters. The potential applications and advancements of LayoutGAN are also discussed, highlighting its significance in the field of graphic design.

Introduction

Layout is an important aspect of graphic design and scene generation. It refers to the arrangement of elements in a visual composition. An accurate layout is critical for good design, as it can help to improve readability, visual appeal, and overall usability.

Traditionally, the layout has been created manually by graphic designers. However, with the advent of deep learning, it is now possible to generate layouts automatically. This is where LayoutGAN comes in. LayoutGAN is a novel generative adversarial network (GAN) that can be used to generate realistic and visually appealing layouts.

LayoutGAN utilizes a multi-step process to generate layouts. It works by first generating a set of randomly placed 2D graphic elements. These elements are then refined using a self-attention module, which helps to ensure that they are aligned correctly. Finally, a differentiable wireframe rendering layer is used to map the generated layout to a wireframe image. A CNN-based discriminator then uses this image to determine whether or not the layout is realistic.

LayoutGAN is effective in generating realistic and visually appealing layouts. It has been used to generate layouts for a variety of applications, including document layout, clipart abstract scene generation, and tangram graphic design.

This report will provide a detailed overview of LayoutGAN and how we reproduced it with modified hyperparameters. We will also discuss the architecture of the network, the training procedure, and the experimental results. We will also discuss the potential applications of LayoutGAN, showcasing its significance in the field of graphic design.

Dataset

The LayoutGAN architecture requires training on suitable datasets to learn the underlying patterns and structures of layouts. Two datasets commonly used in layout analysis and scene generation tasks are the MNIST dataset and the PubLayNet dataset.

The MNIST dataset, a widely recognized benchmark dataset, is a large database of handwritten digits that is commonly used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The MNIST database contains 60,000 training images and 10,000 testing images, providing a diverse range of digit representations.

PubLayNet is a dataset for document layout analysis. It contains images of research papers and articles and annotations for various elements on a page such as “text”, “list”, “figure”, “Title”, and “Table” in these research paper images. The dataset was created by IBM and is available for download from the IBM Data Asset eXchange. It contains over 360,000 document images, each of which is annotated with bounding boxes and polygonal segmentations for the above-mentioned layout elements.

By utilizing these datasets, we aim to train LayoutGAN to generate realistic and visually appealing layouts for both handwritten digits and research paper layouts. The combination of the diverse MNIST dataset and the comprehensive

PubLayNet dataset enables our model to capture the intricacies of different layout types, facilitating the generation of high-quality layouts in various applications.

Literature Survey

The paper "LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators" by Li et al. (2019) introduces a novel generative adversarial network (GAN) that can be used to generate realistic and visually appealing layouts. LayoutGAN works by first generating a set of randomly placed 2D graphic elements. These elements are then refined using a self-attention module, which helps to ensure that they are aligned correctly. Finally, a differentiable wireframe rendering layer is used to map the generated layout to a wireframe image. A CNN-based discriminator then uses this image to determine whether or not the layout is realistic.

The generator consists of two main components: a layout generator and a refiner. The layout generator takes as input a random noise vector and generates a set of randomly placed 2D graphic elements. The refiner then takes the output of the layout generator and refines it by adjusting the positions, sizes, and orientations of the graphic elements. This helps to ensure that the generated layout is aligned correctly.

The discriminator consists of two main components: a wireframe discriminator and a color discriminator. The wireframe discriminator takes as input a wireframe image and determines whether or not it is realistic. The color discriminator takes as input a color image and determines whether or not it is realistic.

LayoutGAN is trained using the adversarial training framework. The generator is trained to generate layouts that are indistinguishable from real layouts, while the discriminator is trained to distinguish between real and fake layouts.

New LayoutGAN: Hierarchical Reinforcement Learning for Furniture Layout in Virtual Indoor Scenes by Yu et al. (2020) is an improved version of LayoutGAN that introduces a hierarchical generative model that can generate layouts at multiple levels of detail. This allows New LayoutGAN to generate more complex and realistic layouts than LayoutGAN.

The hierarchical generative model in New LayoutGAN consists of two main components: a coarse-grained generator and a fine-grained generator. The coarse-grained generator generates a coarse layout, which is a rough sketch of the layout. The fine-grained generator then takes the output of the coarse-grained generator and refines it to generate a detailed layout.

The coarse-grained generator is a convolutional neural network (CNN) that takes as input a random noise vector and generates a coarse layout. The coarse layout is represented as a set of 2D graphic elements, such as rectangles, circles, and lines.

The fine-grained generator is also a CNN that takes as input the output of the coarse-grained generator and generates a detailed layout. The detailed layout is represented as a set of 2D graphic elements, as well as their positions, sizes, and orientations.

LayoutGMN: A Layout Generation Model with Improved Diversity and Control by Zhang et al. (2021) is the latest version of LayoutGAN. The architectural difference between LayoutGMN and LayoutGAN by Yu et al. (2020) is the discriminator. LayoutGMN uses a new discriminator that is more sensitive to small differences in layout. This means that it is better at distinguishing between real and fake layouts. As a result, LayoutGMN can generate more diverse layouts than LayoutGAN by Yu et al. (2020).

The discriminator in LayoutGMN is a multi-scale discriminator. This means that it has multiple layers, each of which is responsible for detecting different levels of detail in the layout. The first layer is responsible for detecting large-scale features, such as the overall shape of the layout. The second layer is responsible for detecting medium-scale features, such as the arrangement of objects in the layout. The third layer is responsible for detecting small-scale features, such as the texture of the objects in the layout.

The use of a multi-scale discriminator allows LayoutGMN to be more sensitive to small differences

in layout. This is because the discriminator can detect small-scale features that are not visible to the human eye. As a result, LayoutGMN can generate more realistic and diverse layouts than New LayoutGAN by Yu et al. (2020).

In addition to the use of a multi-scale discriminator, LayoutGMN also uses a new training algorithm that converges faster. This is due to the use of a new loss function that is more effective at training the discriminator. As a result, LayoutGMN can be trained in less time than LayoutGAN by Yu et al. (2020).

While supervised learning with CNNs has seen widespread adoption in computer vision applications, unsupervised learning with CNNs has received less attention. Soumith Chintala et al.(2016) aim to bridge this gap by proposing DCGANs (DEEP CONVOLUTIONAL GANs), which have specific architectural constraints that make them stable to train in most settings.

They demonstrate the effectiveness of DCGANs by training them on various image datasets. They show that DCGANs learn a hierarchy of representations from object parts to scenes in both the generator and discriminator networks. They also visualize the filters learned by DCGANs and provide empirical evidence that specific filters have learned to draw specific objects.

Furthermore, the authors utilize the learned features from DCGANs for novel tasks, showcasing their applicability as general image representations. They use the trained discriminators for image classification tasks and achieve competitive performance compared to other unsupervised algorithms.

One noteworthy aspect of DCGANs is that the generators exhibit interesting vector arithmetic properties, enabling easy manipulation of many semantic qualities of generated samples. This allows for intuitive control over the generated outputs, making DCGANs versatile in generating images with desired characteristics.

Methodology & Architecture

The LayoutGAN architecture consists of a generator network and two discriminator networks. The generator network takes a random noise vector and a textual description of the layout as input and produces a graphical layout as output. The discriminator networks are used to distinguish between real and fake graphical layouts. The first discriminator network called the region discriminator, focuses on the high-level layout of the image and tries to distinguish between the real and fake layouts. The second discriminator network called the wireframe discriminator, focuses on the low-level structure of the image and tries to distinguish between the real and fake wireframes.

The generator network consists of three parts: an encoder, a visual feature extractor, and a layout decoder. The encoder encodes the description of the layout into a latent representation. The visual feature extractor extracts the visual features from the input image and generates a set of visual features that are used to condition the generator. The layout decoder takes the latent representation and visual features as input and generates a graphical layout.

The wireframe discriminator of LayoutGAN is a Convolutional Neural Network that takes as input a wireframe image and determines whether or not it is realistic. The wireframe discriminator consists of a stack of convolutional layers that learn to distinguish between real and fake wireframe images.

During training, the two discriminator networks are trained to distinguish between real and fake layouts. The generator network is trained to fool the discriminator networks by generating realistic graphical layouts. The training process continues until the generator network can generate diverse and visually appealing layouts that are difficult for the discriminator networks to distinguish from real layouts.

Both Discriminator and Generator, consist of many convolution layers shown in Fig 1, followed by Batch Normalization and Dense layers. The Discriminator layer outputs prediction i.e. whether the generated image is real or fake. The generator layer outputs the

probability of class and geometric parameters for that class. The model consists of both a generator and discriminator using Adam Optimizer with exponential weight decay. We are calculating Generator and Discriminator loss using sigmoid cross entropy loss with logits function.

For the MNIST Dataset, the total number of trainable parameters for the Generator model is 2606850 and the total number of trainable parameters for the Discriminator model is 578625.

We Trained the LayoutGAN for the PubLayNet Dataset which has 9827529 trainable parameters for the Generator model and 2810049 trainable parameters for the Discriminator model.

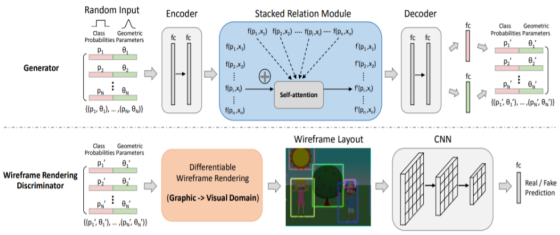

Fig 1. Model Architecture.

Loss functions:

We used sigmoid_cross_entropy_with_logits for the training of the discriminator and Generator. The sigmoid cross-entropy loss also called binary cross-entropy loss is widely used for multi-label classification and can be derived as follows,

$$CE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1-t_1) log(1-f(s_1))$$

Where $f(s_i)$ is a sigmoid function that takes logits as an input and the result of this function will be input for cross entropy function which results in computation loss.

Generator loss: The loss function takes a noise vector as input and generates fake data using the generator model. The discriminator model is then used to classify the generated fake data as real or fake. The loss is calculated using the sigmoid cross-entropy loss function, which measures the difference between the predicted output and the true output.

Discriminator loss: The function then calculates two independent loss values: Real loss and Fake loss. The Real loss represents the loss of the discriminator when it tries to correctly classify real data samples as real, and Fake loss represents the loss when the loss tries to correctly classify fake data samples as fake. The discriminator loss is the total of real loss and fake loss values. The loss is calculated using the sigmoid cross-entropy loss function.

Result

We experimented with different hyperparameters, including changing the optimizer, activation functions, weight decay value, adding/removing convolution layers, and learning rate. After a thorough evaluation, we identified the best-performing model for both the Mnist and PubLayNet datasets.

For Mnist Dataset, At Epoch 0 Discriminator loss is around 1.5784 and the generator loss is around 0.8055 at the learning rate of 0.00001 and the result is visible in (1) of Fig 2.1. Now as the epoch increases, Discriminator loss decreases over time, for epoch 10 Discriminator loss is 1.2234, and the generator loss is 0.8496, note that the discriminator loss decreases while there is a very small change in generator loss. We used a step function to regulate loss after every 10 epochs. The new learning rate is 0.000001 at epoch 20 where discriminator loss is 0.7434 and generator loss is 1.2646. The result can be seen in (2) of 2.1 where you can figure out what the number is in comparison with linear points in Fig 2.1(1).

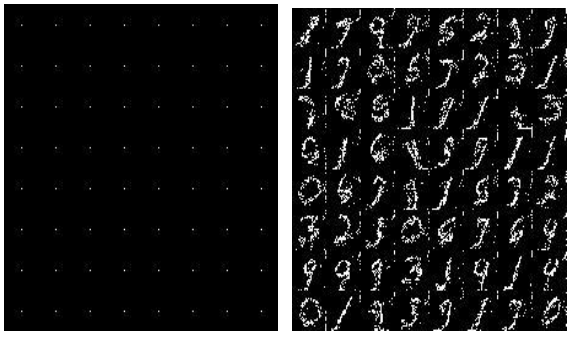

Fig 2.1(1)Mnist Data initial Stage at Epoch 0
(2)Mnist Generator Image After 50th Epoch

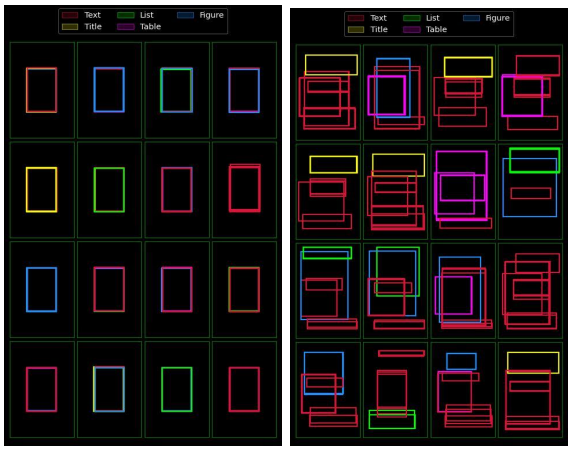

Fig 2.2 (1) Initial Stage of PubLayNet Dataset, where all Classes are at the same place
2) Generated Image after epoch 50

We used the same model architecture with small changes in the input layer for pubLayNet Dataset. Epoch 0 Discriminator loss is 1.3791 and generator loss is around 0.7167 at learning rate 0.00001and Result is visible in (1) of fig 2.1. Generator loss increases and Discriminator loss decreases over time for PubLayNet, Overall Loss can be seen in Fig 3.2. At the End of Epoch 50, we got 0.0710 discriminator loss and 3.9789 generator loss. In Fig 3.2 (2) we can see that the Title layout is on the top if there is any in the dataset and it is almost separated from other layouts.

We are Evaluating Our Model Based on Discriminator Loss, Generator loss, and comparison of the resultant image after each epoch as Predicting the **accuracy** of GANs poses challenges due to several factors. GANs generate samples in a continuous, high-dimensional space, making it difficult to define discrete labels for comparison. Additionally, GANs aim to capture the underlying data distribution, rather than replicating specific instances, making a single **accuracy** metric less meaningful.

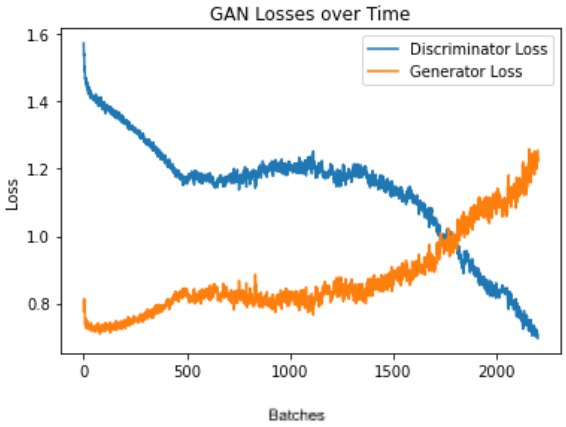

Fig 3.1 GAN Loss Over Time For Mnist Dataset

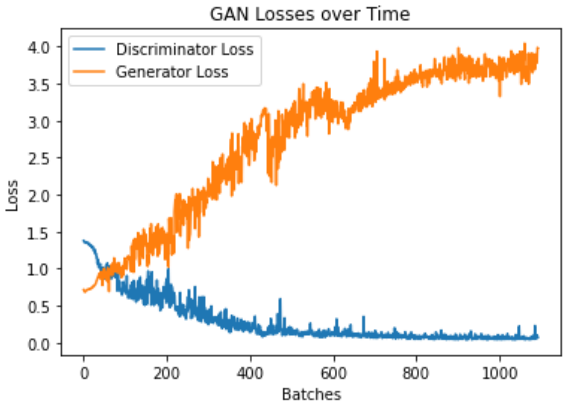

Fig 2.2 GAN Loss Over Time For PubLayNet Dataset

References

Technical Paper
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. Communications of the ACM, 63(11), pp.139-144.

ArXiv Paper
Li, J., Yang, J., Hertzmann, A., Zhang, J. and Xu, T., 2019. Layoutan: Generating graphic layouts with wireframe discriminators. arXiv preprint arXiv:1901.06767.

ArXiv Paper
Zhou, M., Xu, C., Ma, Y., Ge, T., Jiang, Y. and Xu, W., 2022. Composition-aware Graphic Layout GAN for Visual-textual Presentation Designs. arXiv preprint arXiv:2205.00303.

Technical Paper
Patil, A.G., Li, M., Fisher, M., Savva, M. and Zhang, H., 2021. Layoutgmn: Neural graph matching for structural layout similarity. In Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11048-11057).

ArXiv Paper

Di, X. and Yu, P., 2022. Hierarchical Reinforcement Learning for Furniture Layout in Virtual Indoor Scenes. arXiv preprint arXiv:2210.10431.

ArXiv Paper

Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434* (2015).

Technical Paper

Viola, J., Chen, Y. and Wang, J., 2021. FaultFace: Deep convolutional generative adversarial network (DCGAN) based ball-bearing failure detection method. Information Sciences, 542, pp.195-211.

Github repository

https://github.com/JiananLi2016/LayoutGAN-Tensorflow

Github repository

https://github.com/carpedm20/DCGAN-tensorflow

Large Language Model:

https://chat.openai.com/

Large Language Model:

https://bard.google.com/