Fashion image generation using Conditional Deep Convolutional Generative Adversarial Network based on text input

Mosarrat Rumman, Abu Nayeem Tasneem, Israt Jahan Ritun and Annajiat Alim Rasel

August 31, 2021

Fashion image generation using Conditional Deep Convolutional Generative Adversarial Network based on text input

Abstract—**Generative Adversarial Network (GAN) is increasingly becoming a popular research area for generating highly realistic data. However one of the pitfalls of GAN is that it cannot be controlled what a conventional GAN will generate. Therefore we extended a Deep Convolutional Generative Adversarial Network (DCGAN) and added conditioning to it, to generate targeted images according to the user input. We used the fashion MNIST dataset to train our generative model. The text given by the user is matched with predefined fashion categories using fuzzy string matching and the model will generate absolutely new fashion images in accordance with the matched fashion category. The inception score of the generated images are analyzed to evaluate the performance of the model.**

Index Terms—**DNN, Deep Convolutional Network, Generative Adversarial Network**

I. Introduction

Image generation or image synthesis is one of the most common research areas in Machine Learning. Image generation refers to the generation of new images from an existing dataset. During research, limited availability and quality of training data often act as a bottleneck. No matter how powerful a model is or how many computational resources are available to train it, the model's final performance will suffer if the training data does not accurately represent the data distribution that the model is expected to cover [1]. This is why generative models are becoming increasingly popular in the research field.

To generate any kind of data such as images, text or video with machine learning, we need to use a generative algorithm. Using a model to generate new examples that are plausible to derive from an existing distribution of samples, such as generating new images that are comparable but distinct from a dataset of existing photographs, is referred to as generative modeling. Currently, Generative Adversarial Networks (or GANs) are the most promising generative algorithms for image generation.

Generative Adversarial Networks (or GANs) are introduced in 2014 by the famous AI researcher, then a Ph.D. fellow at the University of Montreal, Ian J. Goodfellow and co-author, giving machines the gift of imagination. GAN is an unsupervised machine learning (ML) model in which two neural networks compete to improve their prediction accuracy [2]. They've obtained outstanding results in generating examples for Image Datasets, generating realistic photographs, image-to-image synthesis, text-to-image synthesis, and image super-resolution, among other challenges [3].

Despite the fact that GAN models may produce new random believable instances for a given dataset, there is no method to regulate the types of images that are generated other than attempting to grasp the intricate relationship between the latent space input to the generator and the generated images. The conditional generative adversarial network, or cGAN for short, is a GAN that uses a generator model to conditionally generate images. If a class label is available, image generation can be conditional on it, allowing for the targeted generation of images of a specific type [4]. Many strides have been made in the design and training of GAN models, the most notable of which is the deep convolutional GAN, or DCGAN for short, which specifies the model configuration and training processes that consistently result in the stable training of GAN models for a number of contexts [5].

Generative Adversarial Networks (or GANs) and it's application field is vast. Within a very short time, It has already brought revolution in the most possible Machine Learning fields. Using GANs to produce new probable instances for the MNIST handwritten digit dataset, the CIFAR-10 small object photograph dataset, and the Toronto Face Database was the application described in the original paper by Ian Goodfellow, et al. in the 2014 paper "Generative Adversarial Networks" [2]. Phillip Isola et al. demonstrate GANs, specifically their pix2pix technique, for numerous image-to-image translation tasks in their 2016 publication "Image-to-Image Translation using Conditional Adversarial Networks" [6]. Jun-Yan Zhu presents their famous CycleGAN and a set of highly remarkable image-to-image translation examples in their 2017 publication titled "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks" [7].

In Text to Image generation field, Han Zhang et al. illustrate the use of GANs, specifically their StackGAN, to generate realistic looking photographs from textual documents of simple objects like birds and flowers in their 2016 paper "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks" [8]. Moreover, In their 2016 study "Learning What and Where to Draw," Scott Reed et al. extend on this capability by using GANs to both generate images from text and use bounding boxes and key points as recommendations for where to draw a described object, such as a bird [9]. In their 2016 study "Pixel-Level Domain Transfer," Donggeun Yoo et al. illustrate the use of GANs to generate photos of fashion that may be found in a catalog or online store, based on photographs of models

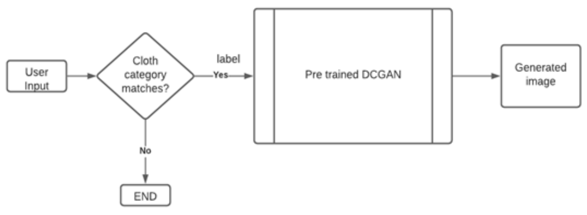

Fig. 1. Overview of the proposed model

wearing the item [10].

In this paper we used a conditional Deep Convolutional Generative Adversarial Network (DCGAN) to generate fashion images based on user input. The contribution of this paper are as follows:

- Match user text input with cloth category using Levenshtein Distance
- Generate absolutely new cloth image that is visually acceptable according to the text category using DCGAN
- Targeted Image Generation.
- Generation of pre-trained model that can be reused.

II. Dataset

In this paper, we used the Fashion MNIST dataset (https://github.com/zalandoresearch/fashion-mnist) which contains a training set of 60,000 and a test set of 10,000 fashion images. Each image is 784 pixels with a height of 28 pixels and a width of 28 pixels. Both the training and test sets have 785 columns along with a label column. The cloth categories are labelled as: T-shirt/top - 0 , Trouser - 1, Pullover - 2, Dress - 3,Coat - 4, Sandal- 5, Shirt - 6,Sneaker - 7,Bag - 8,Ankle boot - 9. The images are black and white so an additional channel dimension is added so that it is compatible with the convolutional layers of the proposed model. Then the pixel values are scaled between -1 to 1 as the generator will output in this range.
In this paper, we used the Fashion MNIST dataset (https://github.com/zalandoresearch/fashion-mnist) which contains a training set of 60,000 and a test set of 10,000 fashion images. Each image is 784 pixels with a height of 28 pixels and a width of 28 pixels. Both the training and test sets have 785 columns along with a label column. The cloth categories are labelled as: T-shirt/top - 0 , Trouser - 1, Pullover - 2, Dress - 3,Coat - 4, Sandal- 5, Shirt - 6,Sneaker - 7,Bag - 8,Ankle boot - 9. The images are black and white so an additional channel dimension is added so that it is compatible with the convolutional layers of the proposed model. Then the pixel values are scaled between -1 to 1 as the generator will output in this range.

III. Methodology

A. Input text processing

The user input text is matched with the pre-defined fashion categories using fuzzy string matching. This will allow slightly differently spelled words to be recognised and matched. We

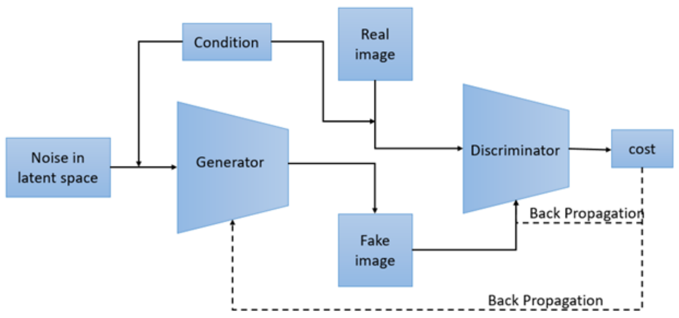

Fig. 2. Architecture of conditional GAN

used Levenshtein Distance [11] to find the similarity ratio between two sequences. Lavenshtein distance is a measure of the number of edit operations needed to change one sequence to another.

$$\mathrm{lev}_{x,y}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} \mathrm{lev}\ _{x,y}(i-1,j)+1 \\ \mathrm{lev}\ _{x,y}(i,j-1)+1 \\ \mathrm{lev}\ _{x,y}(i-1,j-1)+1_{(x_i \neq y_j)} \end{cases} \end{cases} \tag{1}$$

The above formula (1) is used to measure the Lavenshtein distance between the user input x and predefined cloth category y. The 3 functions correspond to deletion, insertion and substitution function orderly. The Levenshtein similarity ratio can be calculated using formula (2).

$$\frac{(|x|+|y|)-\mathrm{lev}_{x,y}(i,j)}{|x|+|y|} \tag{2}$$

Where $|x|$ and $|y|$ are lengths of the texts. In our paper we used a cutoff value of 70%and the category with the highest similarity ratio is selected.

B. DCGAN architecture

In this section the whole architecture of the conditional DCGAN will be described. We built a DCGAN similar to [5] but added condition to it. It has two components- the generator (G) and the discriminator (D). The generator takes input from the latent space [12], i.e, some random variables using Gaussian Distribution and the label of cloth category. G produces some images i according to the label. The discriminator, which is trained with real images along with their labels, classifies the i as real or fake. The classification error of D is then used to update both D and G. The overall process is shown in figure [2]

For building the discriminator model D, a convolutional neural network was modelled with LeakyRelu as activation function. The input to the model are 28x28 gray scale images and an integer representing the label of image (cloth category). The label is passed to the D through an embedding layer and reshaped into 28x28 and concatenated with the input images.

This creates a two channel input image to be passed on to the convolutional layers. We used 2 by 2 stride for downsampling and stochastic gradient descent with a learning rate of 0.0002 and a momentum of 0.5 for optimizing.

The generator model G is also a deep convolutional neural network. For input it takes a random variable from the latent space generated by Gaussian Distribution, as well as the label, passed through an embedding layer in a similar way as described for D. G uses LeakyRelu as activation in the fully connected layers and a hyperbolic tangent as activation in the output layer. The output is a 28x28 image generated based on the label.

Both G and D are combined to form the condition DCGAN, and then trained. We used a sequential model with an adam optimizer and binary cross entropy for calculating the loss. The training occurs in two phase:

- Phase 1: G creates some images from noise in latent space which are forwarded to D.D is trained on real images. D learns to classify real and fake images. The loss is back propagated and the the weights of D are updated The generator is not updated or trained in this phase.
- Phase 2: G produces a batch of images which are fed to D.D does not have the real images this time. D classifies as real or fake.The loss of D is back propagated to G. The weights of G are updated.But the discriminator weight remains unchanged.

The model is trained with 150 epochs with a batch size of 192. These parameters were selected due to better image generation after trying training the model several times with different hyperparameter tuning.. We used google colab GPU and it took approximately 2.5 hours to train the model. After training we saved the model as a pre-trained model.

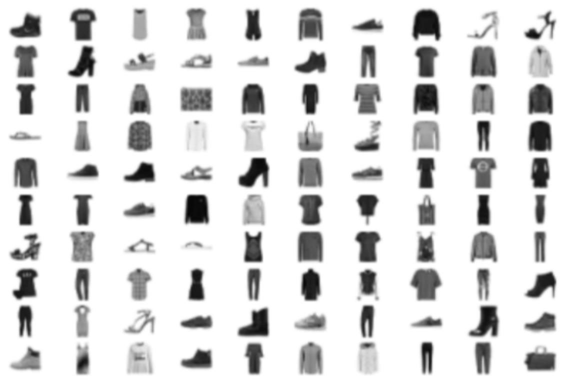
Fig. 3. Original Fashion MNIST images

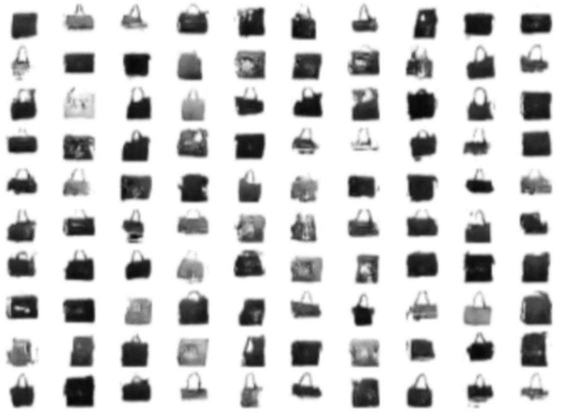
Fig. 4. Generated Fashion images

C. Loss of condition DCGAN

In the conditional DCGAN model, the generator tries to minimize the loss function in equation (3) and the discriminator tries to maximize it [13].

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}}[\log D(\boldsymbol{x} \mid \boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}}[\log(1 - D(G(\boldsymbol{z} \mid \boldsymbol{y})))] \tag{3}$$

Here, y is the label. D($x|y$) is the discriminator output for real data x and D(G($z|y$))] is the discriminator output for generated fake data by G. Ex is the expected value over real images and Ez is the expected value over all generated fake images G($z|y$).

IV. Results

The original images are shown in Figure 3 and the generated images are shown in Figure 4. These images below shows the visual difference of the generated images with the original images, to human eye.

From the given example it can be said that the generated images closely resemble the original image. Even though the images bear close resemblance to the original images, qualitative analysis gives better insight on how the model actually performed. For qualitative analysis of our generated images we used inception score. This was proposed by Salimans, et al. [13] [14] in 2016. It is a popular metric for evaluating images generated by generative models. Inception score is a measure of how realistic the output of a generated model is. As we are generating images of multiple classes, we used the inception model to predict which class the generated image belongs to. The formula for calculating inception score is mentioned in equation (4)

$$\text{I} = \exp\left(\mathbb{E}_{\boldsymbol{x}} \text{KL}(p(y \mid \boldsymbol{x}) \| p(y))\right) \tag{4}$$

Here x denotes the generated sample and y the label predicted by the Inception model. KL is Kullback-Leibler divergence which is basically the conditional probability multiplied by the log of the conditional probability minus the log of the marginal probability.

A pre-trained model was used for the purpose of classification. The model was trained on the training set, to detect the 10 classes of fashion images. Afterwards softmax was used to determine the probability distribution of the predicted classes. 100 images of each class were produced, then the pre-trained model was used to calculate the confidence score of which class the image belongs to. Inception score evaluates the diversity and resemblance of the images to the images the model was trained on.If both of these characteristics are true, then the score will be high.

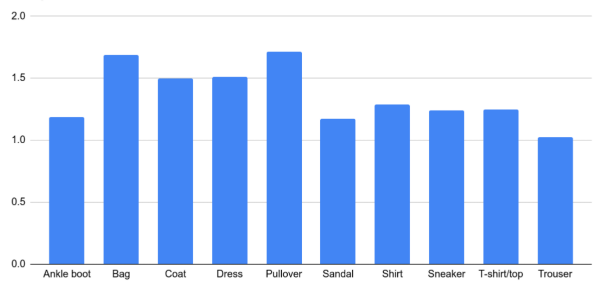

Fig. 5. Inception score for all class labels

From the above bar chart we can see that images of most generated classes scored above 1 in a scale of 2. Which indicates overall acceptable quality of the generated images.

V. Findings

One of the issues we faced with inception score is that it performs well with colored images only. As the images generated by our model are greyscale, the score had high confidence for multiple items, in case of fashion items that look similar to each other. Also for grayscale image output we couldn't use the Inception V3 as it works only on colored images.

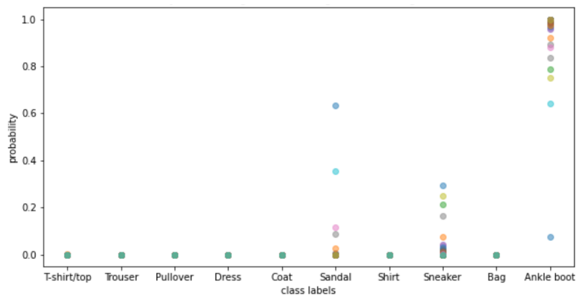

Fig. 6. Probability distribution given that, generated image is Ankle Boot

Above image is the probability distribution for the "Ankle boot" class. We can see, the score of "Ankle boot" class is the highest. Also, similar looking shoe items also had high scores compared to other fashion items. This shows that the classification algorithm found ankle boot is similar to other shoe items, which is a result of poor discriminatory factors.

VI. Conclusion

Generative models require very high computation power and a huge amount of time to produce satisfying results. For lack of computing power we had to use grey scale images and images of very small resolution. Even though our model produced acceptable results, it can be improved in various ways. With more computing power and time, it is possible to train a model which is able to generate better looking and higher resolution images. We only used syntactic analysis of Language to detect the desired class of item to generate. Also, in future, complex language understanding can be used to do contextual analysis and make a model which is able to understand inputs like "Red T-Shirt with long sleeve" and produce respective output. Extensive hyper-parameter tuning can improve the quality of generated images. Finally yet importantly, more complex generative architectures, such as, CYCLE-GAN, ATTENTION-GAN [13], etc can be used to improve the model even further.

References

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first . . ."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

References

[1] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B. and Bharath, A.A., 2018. Generative adversarial networks: An overview. IEEE Signal Processing Magazine, 35(1), pp.53-65.

[2] I. J. Goodfellow et al., "Generative Adversarial Networks," arXiv.org, 2014. https://arxiv.org/abs/1406.2661.

[3] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," arXiv:1805.08318 [cs, stat], Jun. 2019, [Online]. Available: https://arxiv.org/abs/1805.08318.

[4] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," arXiv.org, 2014. https://arxiv.org/abs/1411.1784.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," arXiv.org, 2015. https://arxiv.org/abs/1511.06434.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and Efros, Alexei A, "Image-to-Image Translation with Conditional Adversarial Networks," arXiv.org, 2016. https://arxiv.org/abs/1611.07004.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," arXiv.org, 2017. https://arxiv.org/abs/1703.10593.

[8] H. Zhang et al., "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," arXiv.org, 2016. https://arxiv.org/abs/1612.03242.

[9] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning What and Where to Draw," arXiv:1610.02454 [cs], Oct. 2016, [Online]. Available: https://arxiv.org/abs/1610.02454.

[10] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, "Pixel-Level Domain Transfer," arXiv:1603.07442 [cs], Nov. 2016, Accessed: Aug. 31, 2021. [Online]. Available: https://arxiv.org/abs/1603.07442.

[11] S. Rani and J. Singh, “Enhancing Levenshtein’s Edit Distance Algorithm for Evaluating Document Similarity,” Communications in Computer and Information Science, pp. 72–80, 2018, doi: 10.1007/978-981-13-0755-36.

[12] T. P. Van et al., “Interpreting the Latent Space of Generative Adversarial Networks using Supervised Learning,” 2020 International Conference on Advanced Computing and Applications (ACOMP), pp. 49–54, Nov. 2020, doi: 10.1109/ACOMP50827.2020.00015.

[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved Techniques for Training GANs,” arXiv.org, 2016. https://arxiv.org/abs/1606.03498.

[14] K. Shmelkov, C. Schmid, and K. Alahari, “How good is my GAN?,” arXiv:1807.09499 [cs], Jul. 2018, Accessed: Aug. 31, 2021. [Online]. Available: https://arxiv.org/abs/1807.09499.

[15] X. Chen, C. Xu, X. Yang, and D. Tao, “Attention-GAN for Object Transfiguration in Wild Images,” arXiv:1803.06798 [cs], Mar. 2018, Accessed: Aug. 31, 2021. [Online]. Available: https://arxiv.org/abs/1803.06798.