Deep Learning Fashion Lookbook: Spring 2020
CS230 Final Report

Aishwarya Rameshkumar
Stanford University, Department of Computer Science
Project Category: Generative Modeling Application
aishrk@stanford.edu

Abstract

The fashion and design industry is a globally-encompassing $2.4 trillion dollar industry which has been around for centuries; however, throughout this time, the traditional approach to design with human designers has largely stayed the same. With recent advancements, we now have the opportunity to utilize strides in deep learning to rethink the approach to fashion design and generation. This project explore the use of DCGANs as well as other deep learning techniques such as ANN Classifiers in the exploration and generation of new fashion outfit looks. The final deliverable is an end-to-end pipeline including GAN/Classifier models, analysis modules, and more which encompasses the entire machine learning cycle and results in generated clothing images near state-of-the-art image quality and the generation of a new model-based fashion outfit for a lookbook.

1 Introduction

The fashion and design industry is a globally-encompassing $2.4 trillion dollar industry which has been around for centuries [1]. Over this time, however, the traditional approach to fashion design has largely stayed the same, with human designers finding inspiration and creating new designs independently [2]. With the recent technological advancements, there is an opportunity to utilize strides in machine learning and deep learning to revolutionize the industry.

This project aims to explore the use of GANs, or Generative Adversarial Networks, in the exploration and generation of new fashion looks through computer generated means. The primary goal is to be able to utilize GANs and/or additional deep learning and heuristic based techniques to be able to create and recommend new outfits for a fashion lookbook based on generated clothing images. The final deliverable will be an end-to-end pipeline including GAN models, classifiers, analysis modules, and more which encompasses the entire machine learning cycle from data processing/wrangling, hyperparameter tuning, model training and testing, generation, classification, prediction, analysis, experimental iteration, and a final outputted model and outfit. The final GAN-generated outfit images get very close to state-of-the-art images with respect to image quality [3] and the classifier is trained to an accuracy of [Train: 96.17%, Test: 85.71%]. The paper is organized as follows: (Section 2) Related Work, (Section 3) Dataset and Features, (Section 4) Methods, (Section 5) Experiments/Results/Discussion, and (Section 6) Conclusion/Future Work.

2 Related Work

Previous approaches in the space of applying GANs to the domain of fashion looks/outfit generation largely include the use of Progressive GANs (P-GANs) and StackGANs [3]; however, these models were explored in the context of text based image generation and also used datasets that have significant differences from the DeepFashion2 dataset currently being used for this project. Other existing approaches leverage the idea of conditional GANs to create fashion recommendations conditioned on a certain attribute of the input [4]. Separate works have also been done with respect to classification of clothing using CNNs [13]. While each of these works are tailored to certain scenarios, the conceptual ideas, corresponding datasets, and the applicability of these works was still explored over the course of this project. However, the final approach for this project differs/has novelty from known related work in the use of the combination of image generation and classification along with an implemented end-to-end pipeline to create a new generated lookbook outfit as described in the following sections.

3 Dataset and Features

For this project, we are using the DeepFashion2 (DF2) dataset. DeepFashion2 is an extensive dataset consisting of both commercial and consumer fashion images spanning numerous clothing categories, where every item in an image includes a label describing the bounding box, landmarks, clothing category, style, scale, occlusion, zoom-in, and viewpoint. Each of these items are described in further detail below [5].

The dataset is split into training, test, and validation sets, and key statistics/attributes of the dataset are summarized in

Figure 1 below. The "images" field represents the number of images in the dataset. The "bounding box" field refers to an attribute of the dataset that provides coordinates [x1,y1,x2,y2] bounding the imagewhere the upper left coordinate is (x1,y1) and the lower right coordinate is (x2, y2). And the "landmarks" field refers to an attribute of the dataset with values [x1,y1,v1,...,xn,yn,vn] that essentially represent key points outlining each of the categories of clothes, where (xn, yn) represent the coordinate of the point and (vn) represents the visibility of the point in the picture (v=2 is visible, v=1 is occlusion, v=0 is unlabeled).

Figure 1: Dataset Key Statistics

	Train	Validation	Test	Total
images	390,884	33,669	67,342	491,895
bounding box	636,624	54,910	109,198	800,732
landmarks	636,624	54,910	109,198	800,732

The 13 clothing categories included in the dataset include: (1) short sleeve top, (2) long sleeve top, (3) short sleeve outwear, (4) long sleeve outwear, (5) vest, (6) sling, (7) shorts, (8) trousers, (9) skirt, (10) short sleeve dress, (11) long sleeve dress, (12) vest dress, and (13) sling dress. A visual of the categories can be found in the Appendix (Figure 19). Figure 2 below shows the distribution of the dataset/images across these categories. The images of the dataset are also described by scale, occlusion, zoom-in, and viewpoint. A visual of these attributes can be found in the Appendix (Figure 20). Figure 3 below shows the distribution of images across these attributes. Scale refers to the same article of clothing shown at different sizes/proportions. Occlusion refers to the same article of clothing shown in different distortions or when partially hidden. Zoom-in refers to the same article of clothing shown at different zoomed-in views. And viewpoint refers to the same article of clothing shown from different angles. Finally, the style refers to the same article of clothing in a different pattern or variation (e.g. in the last row of Appendix (Figure 20), there are 3 styles - green, orange, purple - of the dress). Finally, each image in the dataset also contains its own json file, which describes all of the above attributes including source (i.e. commercial or customer taken image), category, bounding box, landmarks, style, scale, occlusion, zoom-in, viewpoint, etc.

This dataset is very extensive and contains many details about the images and clothing that may be utilized and influence the outcome of the project. Attributes such as the scale, occlusion, zoom-in, and viewpoint (and their distributions) may especially have such implications in the project as they may have significant impact on the final generated image. For example, since 50% of images are of a moderate scale, the generated images may also largely be moderate sized images (as opposed to small or large), and since 73% of images have a frontal viewpoint, we may expect that the majority of our generated images would be in a relatively front facing position. Furthermore, based on the distribution of clothing categories, we may expect to see more items like short sleeve tops and trousers being generated than short sleeve outwear or sling dresses. There may also be a potential to use attributes such as bounding box and landmarks to guide the image generation as well.

In addition to the DeepFashion2 dataset, for the final project milestone, we ended up augmenting the dataset with a new dataset, the Fashion Product Images Dataset (FPID)[16]. This data augmentation ended up being critical for the performance of the DCGAN and Classifier, as will be discussed further in the modelling section. The FPID consists of 44k images described in a corresponding csv with the columns: id, gender, masterCategory, subCategory, articleType, baseColour, season, year, usage, and productDisplayName. The most relevant column for this project being articleType, which describes the actual category of clothing at the targeted granularity (e.g. T-shirts, Dresses, Jeans, etc.). There are a total of 143 categories in articleType of which the top 5 include (1) Tshirts, (2)Shirts, (3)Casual Shoes, (4)Watches, (5)Sports Shoes. The Figure 4 below shows the high level distribution trend of the articleTypes over the 143 categories, and we observe that only 10 of the categories have over 1000 images each in the dataset, which again likely has the implication that these are also the categories of images which will be most generated by the DCGAN.

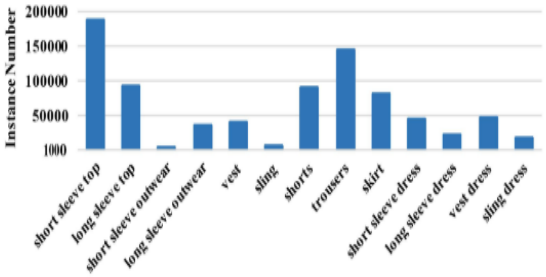


Figure 2: DF2 Clothing Categories Distribution

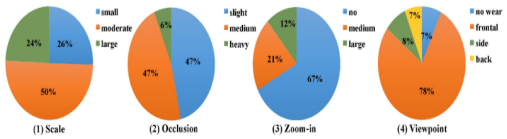


Figure 3: DF2 Clothing Attributes Distribution

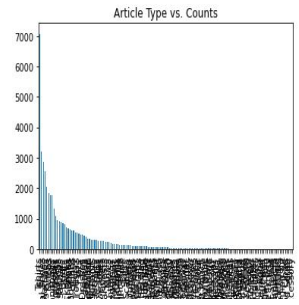
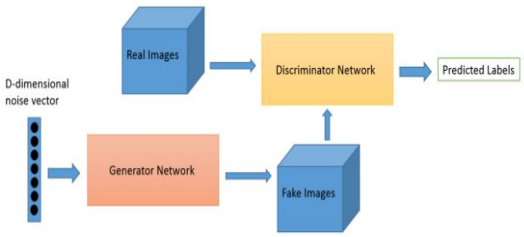


Figure 4: FPID ArticleTypes vs. Counts

4 Methods

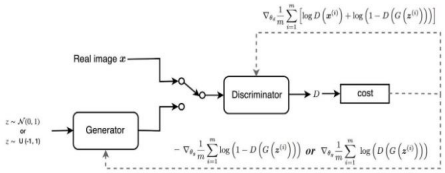
For this project, there were 2 primary deep learning methods used: (1) DCGANs and (2) ANNs. The DCGAN, or Deep Convolutional Generative Adversarial Network, is essentially a GAN which employs CNNs, or Convolutional Neural Networks, for the generator and discriminator components. On a high-level, the GAN is a "generative model with deep neural network, and often applied to the image generation" [14]. The architecture and data flow of a GAN can be found in Figure 5 below. The objective function/loss of the GAN is measured via cross-entropy, with the equations found in Figure 6. Finally, the Deep Convolutional part of the DCGAN comes into play

by the inclusion of CNNs in the GANs generator and discriminator as shown in Figure 7. Some key differences between a DCGAN and a regular GAN also include: (1)replacing max pooling with convolutional stride, (2)using batch normalization except for output/input layers of G/D, (3)using transposed convolution for up/downsampling, etc. More information can be found at [17][18][19][20].



(source: https://www.oreilly.com/ideas/deep-convolutional-generative-adversarial-networks-with-tensorflow)

Figure 5: GAN Data Flow



(source: https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09)

Figure 6: GAN Objective Functions

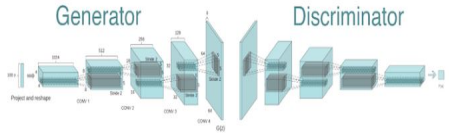


Figure 7: DCGAN - CNN Generator/Discriminator

The ANN, or Artificial Neural Network, has countless applications, but in the context of this project, it was used as a classifier. The model employed included an input layer, hidden layers using the "relu" loss function, and the output multi-class classification layer. Optimizers tested included SGD, RMSProp, and Adam, of which Adam - which "calculates the individual adaptive learning rate for each parameter from estimates of first and second moments of the gradients" - worked the best for this scenario [15]. More information can be found at [21].

5 Experiments/Results/Discussion

5.1 Mileston 1: Baseline Experiments

For the milestone 1 baseline, the environment setup was made using Colab, Tensorflow, and the DeepFashion2 dataset. The model used was a public implementation of the DCGAN used for image generation [6][Code in reference]. This initial model took time to set up as the implementation had issues out of the box working with the dataset and on the colab environment. The model was then trained with varying a small subset of parameters/values (learning rate, batch size) and epochs; one challenge/realization encountered here was that training each model takes a significant amount of time (multiple hours), which constrains the rate of experimentation. Another challenge was building the intuition for tuning, and like we learned in class, it has proved to be an iterative experimental process. The final model used for the baseline consisted of 100 epochs, an alpha learning rate of 0.0004, a beta rate of 0.5, a batch size of 30, and a weight initialization of 0.02. During our discussion, Jo, our Project TA, also mentioned that we should include the loss graphs and outputted image as the evaluation measure/hard results as these measures are more appropriate for GANs. Therefore, the results for the loss (discriminator in blue, generator in orange) and generated image after the last (100th) epoch of the GAN training are included in Figures 8 - 9 below.

Based on the results, after the first epoch the GAN generator and discriminator vary significantly in their loss [D Loss: 7.66456, G Loss: 0.00029] and no discernible image is produced yet (grey image). However, after 100 epochs, we see the GAN generator and discriminator loss have started to converge [D Loss: 1.00700, G Loss: 0.62808] and an image is taking form with discernible outlines (Figure 9). The image produced by after 100 epochs is still noisy and we can't really make out parts of an outfit however, so we will work on improving this as the next step for milestone 2.

5.2 Milestone 2: Further Experimentation and Improvements

For milestone 2, the primary focus was further experimentation and improving the original DCGAN results from milestone 1. As part of the further experimentation to improve the original DCGAN model and produce more identifiable clothing images, we methodically experimented with various aspects of the model and hyperparameters including (1) the optimizer, (2) the generator, (3) the discriminator, (4) generator learning rate, (5) discriminator learning rate, (6) beta learning rate, (7) batch size, (8) epochs, (9) epsilon value, and (10) noise size. For the optimizers, the tensorflow optimizers SGD, RMSProp, and Adam were tested as these are common ones used for GAN tuning [9], of which Adam performed best. For both the generator and discriminator, we also experimented with the convolution layer with respect to kernel size, initialization, and normalization. Finally, with respect to the numerous parameters to tune, the high-level approach we took was to come up with a reasonable upper and lower bound based on community research of GANs and previous random experiments and first attempt a binary search approach with a few values to identify a region which made more sense to fine-tune. We then performed guided experiments based on loss and generated output to further identify the best values for the parameters. Over 40 experiments were performed systematically, taking approximately over 300 hours. Figure 21 in the Appendix summarizes the range of values over which the experiments were conducted, including lower bound, upper bound, and performance notes which describe around which value the performance seemed best. As an aside, we also investigated other DCGAN approaches (e.g. pytorch), however, the original model seemed very similar to other implementations, flushed out, and better structured.

The final results for the best image produced by methodical experimentation is are included in Figures 10 - 11 below. The corresponding final model used for this result consisted of 300 epochs, a generator/discriminator learning rate of 0.0001, a beta rate of

0.3, a batch size of 64, epsilon of 0.00008, and noise size of 90. The values that seemed to have more impact on the performance of the model were by far the number of epochs, then learning rates, epsilon, and batch size. These results are expected with respect to the fact that the learning rates, epsilon, and batch size usually have significant impact on the performance of the model. The results were also expected/made sense from the perspective that the lower learning rates and epsilon rate and higher batch size rate yielded in better performance for this scenario. With respect to noise and epochs, the image quality consistently improved with increasing epochs. However, while it seems from community research that 300 epochs may be enough in some cases to see better quality images, it may be that since there is a very large number of detailed fashion images, it may require even more epochs to see a better image (although it would then also require significantly more training time). One potential next step improvement here could simply be increasing the epochs to a very large number (e.g. 500-1000) - although it may take days to run - to gauge change in performance.

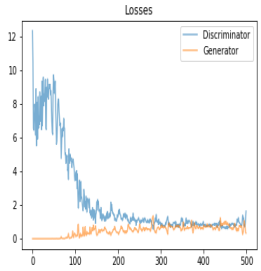


Figure 8: MS1: Loss Epoch 100

Figure 9: MS1: Image Epoch 100

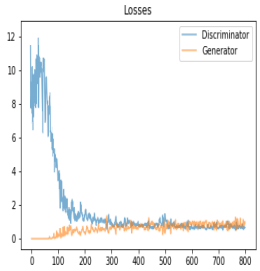


Figure 10: MS2: Loss Epoch 300

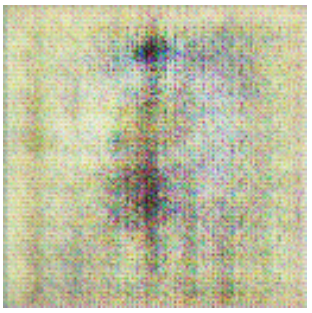


Figure 11: MS2: Image Epoch 300

Compared to the baseline, the final image, while still containing some noise, shows a visible improvement in that we can start to make out a figure with a head area, top outfit area, and bottom outfit area. The GAN generator and discriminator loss have also converged significantly at [D Loss: 0.67167, G Loss: 0.87957]. which is closer than the baseline. With respect to next steps following Milestone 2, as a note, one potential pivot approach that was discussed with TA Jo was to potentially switch from using GANs to using LSTMs/CNNs as a means to recommend a final outfit and still keep somewhat in line with the goal of the project [12]. This option was briefly explored post Milestone 2; however, it was found that it did not work well for the scenario in mind to use a generated image as a means for recommending a new outfit, and the decision was made not to proceed with this method. One of the key critical next steps following Milestone 2 was also to experiment with simply running with a very large number of epochs (500 minimum) and observing results. As discussed in the next section, running these experiments yielded some critical insights towards the improvement/augmentation of the GAN based approach and the success of the project overall.

5.3 Final Milestone: Improved/Augmented Approach, Experimentation, and E2E Pipeline

For the final milestone, we discussed with TA Jo, and the objectives were the following: (1) improve GAN model performance significantly to be able to generate distinguishable clothing items, (2) augment the model/approach in a meaningful way, and (3) create an end-to-end project pipeline that accomplishes the original goal of creating a deep learning fashion lookbook that outputs/recommends new generated outfits.

For the first objective of improving the GAN generated clothing images, we first tried training the GAN model for significantly more epochs (this was the original next step from Milestone 2). Upon training it for 500 epochs on the original DeepFashion2 dataset, more distinguishable images were being generated as in Figure 12 below; however, at this point we observed that even in these images, there seemed to be a lot of noise around the main clothing piece. Then, we realized looking at the original DeepFashion2 dataset that this is likely due to the pictures in the dataset being quite complex (many include a background in addition to the main outfit) as in Figure 13. To address, we pivoted to experiment with augmenting our dataset itself with additional clothing images which (1)focused on the clothing pieces and (2)had little to no background. Hence, we found the FPID dataset which fit this criteria as in Figure 14 [16]. As a final dataset, accounting for the hardware and performance constraints of Google Colab, we then used 40k images [randomly sampled 20k from DeepFashion and 20k from FPID] to train the DCGAN with the same final hyperparameters as from Milestone 2 and 400 epochs. Compared to the Milestone 1 and Milestone 2 baselines, the vast majority of final GAN generated clothing images at the end of training showed significant improvement and were clearly distinguishable articles of clothing. Figures 15 and 16 below show an example of the final loss (quantitative metric) as well as the final generated image (qualitative metric). The final measured metric of loss of the discriminator and generator converged at [D Loss: 0.79576, G Loss: 0.83892]. The Appendix Figures 22 - 31 includes depictions of the progression of DCGAN training over different epochs.

For the second and third objectives, we then primarily thought about what we could add to our approach to help us achieve our goal of creating a deep learning fashion lookbook that outputs/recommends new generated outfits. At this point, we had a set of generated images with no categories and needed a way to eventually convert that into a new outfit for a "lookbook". A natural bridge then would be a way to classify the generated images into categories to later pair them for a new outfit, and in accordance, we augmented our model and approach with (1) an ANN Classifier model and (2) created an end-to-end pipeline around both the DCGAN and ANN Classifier for then processing all inputs, training and testing the models, and generating a new outfit pairing from the GAN-generated clothing images based on a heuristic approach. Figure 17 shows the high-level architecture of this end-to-end pipeline.

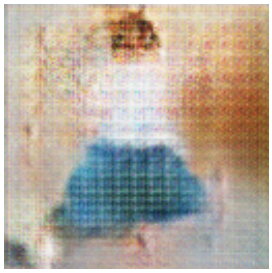


Figure 12: DeepFashion2 Epoch 500

Figure 13: DeepFashion2 Example

Figure 14: FPID Example

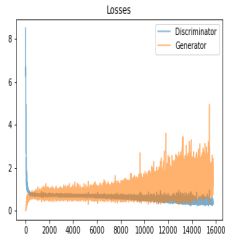


Figure 15: Loss Epoch 400

Figure 16: Image Epoch 400

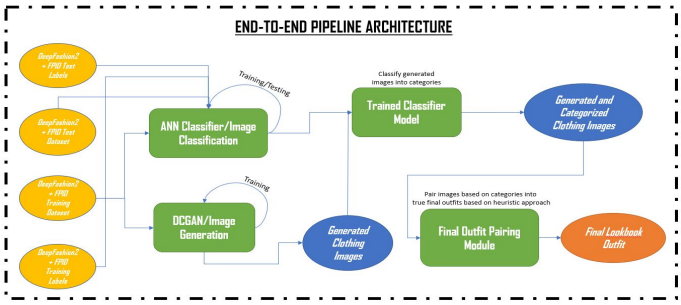


Figure 17: End-to-End Pipeline Architecture

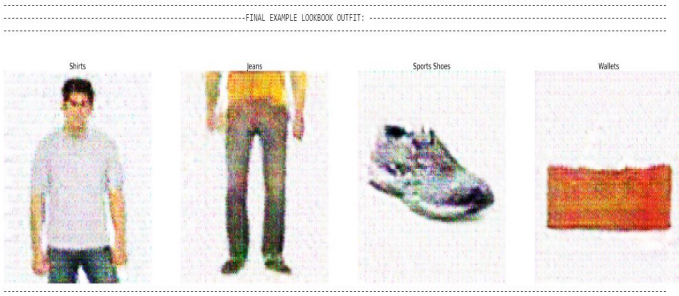


Figure 18: Final Lookbook Outfit Example

We begin by training both the DCGAN Image Generator (explained previously) and the ANN Classifier. For the ANN Classifier, the input was the training dataset (with DeepFashion2 + FPID) and their corresponding category classification labels (e.g. T-shirts, skirt, etc.) as well as a test set consisting of 10k images [randomly sampled 5k from DeepFashion2 and 5k from FPID], which yields the standard [80% train: 20% test] split of data as discussed in class. Various activation functions were tested for the middle layer including tanh, relu, leaky relu, and softmax, and it was found that relu both provided the best training and test accuracy and converged the most quickly (final 400 epochs). Learning rates were also experimented with the Adam optimizer, and while many rates ended up causing the model to be stuck at 0.1743 accuracy, the final learning rate of 0.0001 enabled the model to converge quickly and accurately. The code was self-implemented adapting the tutorial here [23]. The final quantitative metrics for the trained model yielded a [Train Accuracy: 0.9617] and [Test Accuracy: 0.8571]. Additional analysis code was also included to gauge label accuracy and plot the test images against their predicted labels.

Once we have our trained and tested model from the ANN Classifier, we then proceed to train the DCGAN [the only reason for this ordering is because we are using a single notebook in Colab] which outputs the generated images. The generated clothing images are taken as input into the trained classifier model which then predicts labels/clothing categories for each of the GAN-generated clothing images and outputs them as dictionaries. Examples of the GAN-generated images with predicted labels are in the Appendix Figure 32. Finally, a final outfit pairing module takes in the generated images with their category classifications and - using random preset heuristic outfit pairings (e.g. Shirts/Jeans/Sports Shoes/Wallets) - outputs the final outfit for the lookbook with each of the GAN-generated and ANN classified clothing articles as in Figure 18. [Multiple new outfits for the lookbook can be generated by running the last cell in the notebook.] Thereby, we effectively accomplish our goal of creating an end-to-end deep learning fashion system capable of generating new clothing images and outputting an outfit for a lookbook.

6 Conclusion/Future Work

In conclusion, through this project, we have explored the use of GANs to generate new fashion looks through computer-generated means. We have explored various related works; applied deep learning models, approaches, and techniques learned both through class and independently; thoroughly experimented and undergone countless iterations of the full machine learning cycle; and developed a final deliverable through which we accomplish our goal of creating an end-to-end deep learning fashion system capable of generating new clothing images and outputting an outfit for a lookbook. This novel end-to-end pipeline includes DCGAN-generated clothing images, ANN classification, analysis, etc., and, most significantly, a final outputted lookbook outfit whose image quality comes close to state-of-the-art GAN generated fashion images [3].

Future work for this project would include exploring the option of conditioning the DCGAN using embedding layers [4]. This component was actually explored during the course of the final milestone and is already in the process of implementation but was not included here due to time constraints relating to integration of the piece with the existing pipeline. This piece would involve conditioning on singular garment pieces/styles. Additional work could also include the investigation of SRGANs to now make the images even more realistic [22]. We definitely plan on exploring both of these components following project submission as they would be interesting additions to explore. Based on the work done on this project as well as the plethora of additional avenues to explore, it is truly an exciting time for both the deep learning community and the fashion industry to see what lies ahead for technological fashion generation going forward!

7 Contributions

Team Member: Aishwarya Rameshkumar. As this was a one person team, Aishwarya Rameshkumar was responsible for all primary contributions to the project and experimentation/implementation/reports.

TA: Jo Chuang. TA Jo helped with discussing the project milestones and deliverables, brainstorming ideas, and providing general great support, guidance, and advice throughout the course of the project!

8 Appendix

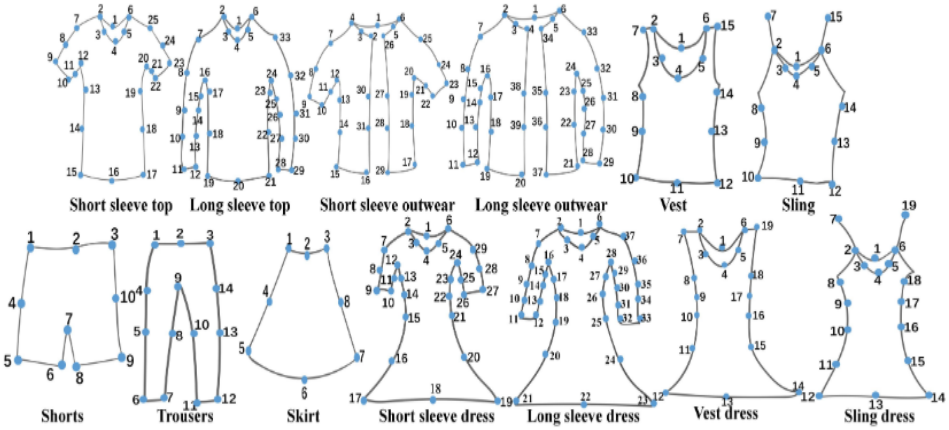


Figure 19: Clothing Categories with Landmarks

Figure 20: Clothing Attributes [Scale, Occlusion, Zoom-In, Viewpoint]

Figure 21: Experiment Parameters Tuning Overview

	Lower Bound	Upper Bound	Region of Improved Performance
Generator Learning Rate	0.00001	2	~0.0001
Discriminator Learning Rate	0.00001	2	~0.0001
Beta Rate	0.00001	2	~0.3
Batch Size	10	128	~64
Epochs	30	300	~300
Epsilon Value	0.000001	0.001	~0.00008
Noise Size	10	150	~90

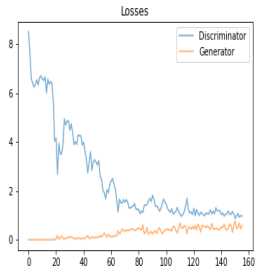


Figure 22: DCGAN Loss Epoch 1

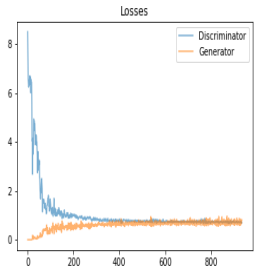


Figure 23: DCGAN Loss Epoch 50

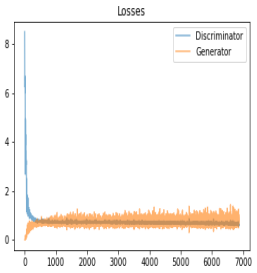


Figure 24: DCGAN Loss Epoch 100

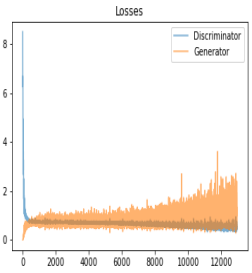


Figure 25: DCGAN Loss Epoch 250

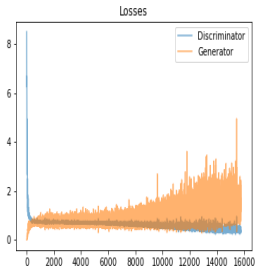


Figure 26: DCGAN Loss Epoch 400

Figure 27: DCGAN Image Epoch 1

Figure 28: DCGAN Image Epoch 50

Figure 29: DCGAN Image Epoch 100

Figure 30: DCGAN Image Epoch 250

Figure 31: DCGAN Image Epoch 400

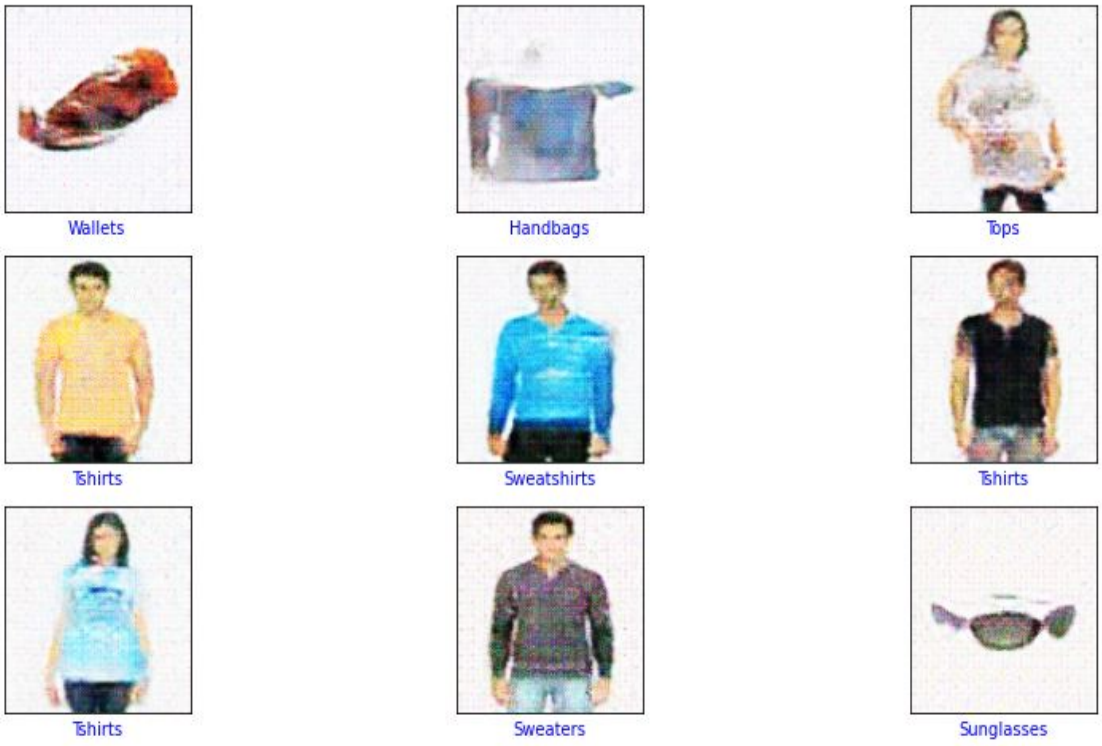


Figure 32: GAN-Generated Images with Predicted Category Labels

9 References

[1]https://www.mckinsey.com/industries/retail/our-insights/the-state-of-fashion
[2]https://en.wikipedia.org/wiki/History_of_fashion_design
[3]https://arxiv.org/pdf/1806.08317.pdf
[4]https://arxiv.org/pdf/1906.05596v1.pdf
[5]https://github.com/switchablenorms/DeepFashion2
[6]https://github.com/gsurma/image_generator
[7]http://openaccess.thecvf.com/content_cvpr_2017/papers/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.pdf
[8]http://openaccess.thecvf.com/content_CVPRW_2019/papers/FFSS-USAD/Zou_FashionAI_A_Hierarchical_Dataset_for_Fashion_Understanding_CVPRW_2019_paper.pdf
[9]https://towardsdatascience.com/understanding-and-optimizing-gans-going-back-to-first-principles-e5df8835ae18
[10]https://arxiv.org/pdf/1905.12862.pdf
[11]https://sagarverma.github.io/others/icip18-fashion-diversity.pdf
[12]https://github.com/sagarverma/FashionRecommendationST-LSTM
[13]https://arxiv.org/pdf/1811.04374.pdf
[14]https://medium.com/@keisukeumezawa/dcgan-generate-the-images-with-deep-convolutinal-gan-55edf947c34b
[15]https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3
[16]https://www.kaggle.com/paramaggarwal/fashion-product-images-dataset
[17]https://medium.com/@jonathan_hui/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f
[18]https://medium.com/@ramyahrgowda/dcgan-implementation-in-keras-explained-e1918fc930ea
[19]https://towardsdatascience.com/image-generator-drawing-cartoons-with-generative-adversarial-networks-45e814ca9b6b
[20]https://medium.com/@jonathan_hui/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09
[21]https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6
[22]https://arxiv.org/abs/1609.04802
[23]https://www.tensorflow.org/tutorials/keras/classification