Using Generative Adversarial Networks to Design Shoes: The Preliminary Steps

Jaime Deverall
Stanford University
jaimedev@stanford.edu

Jiwoo Lee
Stanford University
jlee29@stanford.edu

Miguel Ayala
Stanford University
mayala3@stanford.edu

June 13, 2017

Abstract

In this paper, we envision a Conditional Generative Adversarial Network (CGAN) designed to generate shoes according to an input vector encoding desired features and functional type. Though we do not build the CGAN, we lay the foundation for its completion by exploring 3 areas. Our dataset is the UT-Zap50K dataset, which has 50,025 images of shoes categorized by functional type and with relative attribute comparisons. First, we experiment with several models to build a stable Generative Adversarial Network (GAN) trained on just athletic shoes. Then, we build a classifier based on GoogLeNet that is able to accurately categorize shoe images into their respective functional types. Finally, we explore the possibility of creating a binary classifier for each attribute in our dataset, though we are ultimately limited by the quality of the attribute comparisons provided. The progress made by this study will provide a robust base to create a conditional GAN that generates customized shoe designs.

1 Introduction

1.1 The Shoe Industry

The demand for shoes is greater than ever and it is continuing to grow. By 2023, the market is expected to have a value of $258 billion. [9]. Consequently, the industry has been evolving at a tremendous rate. Part of this involves streamlining the design and production processes. Most of the technological breakthroughs in the shoe industry have revolutionized the production component through automation. However, the design aspect remains a major bottleneck in getting more shoes to market. Currently, shoe designs take root in the mind of human designers and are meticulously refined over many iterations. What we want to see is if we can use the latest advances in artificial intelligence to digitally generate new shoes in order to speed up the process of shoe design.

1.2 Shoes and Neural Networks

Due to the improvement of computer vision architectures, particularly that of Convolutional Neural Networks (CNN), there have been multiple attempts to apply artificial intelligence to the field of fashion and shoes. For instance, we have seen fashion-related convolutional neural network implementations such as shoe recommendation [18], clothing description [1] and fashion apparel detection [14]. However, there seems to be limited work in the area of shoe design generation.

1.3 Procedural Image Generation

Recent breakthroughs in the field of computer vision have led to unprecedented success in procedural image generation. These solutions are able to generate images that are quickly becoming indistinguishable from real images. In recent years, several generative models have been pioneered. For example, both Pixel Recurrent Neural Networks [25] and PixelCNN [26] have been very successful at generating images.

1.4 Generative Adversarial Networks

Another powerful model that has arisen is the GAN [13]. Goodfellow et. al proposed a system for generating images based on a Generator network, G, and a Discriminator network, D. At each training iteration, G generates a set of images and tries to make them as realistic as possible. Simultaneously, D tries to determine whether or not each of G's images are real or not. As both G and D train against each other, the generated images should become more and more realistic. Based on this model, researchers from different countries have developed interesting applications for the technology including image animation [33], image super-resolution [22] and text to image synthesis [29].

Since 2014, several variations on the GAN have appeared with impressive results. One such approach utilized multiple GANs [7] to each generate a different layer of a Laplacian pyramid [2]. The output of each GAN was later combined to produce the final image. Images produced by this method seemed to be much more photorealistic than those created by other models. Another successful modification of the GAN is the Deep Convolutional GAN which yielded results by combining the strengths of CNNs and GANs [28].

If we intend to make a shoe generator that adapts to consumer trends, we will need to be able to input parameters that modify our output. We will be able to do this with a Conditional GAN (CGAN) [24]. With a CGAN, we can feed in data to condition both the Discriminator and Generator on. For example, if we had a CGAN for faces, we could feed in attributes signifying race and age and end up with a photo that reflected these qualities [10]. While there are other shoe generation networks out there [35], there are few that are conditioned on pertinent attributes.

1.5 Problem Statement

For our dream shoe generator, we envision a CGAN that takes in a vector of features signifying the qualities that an individual desires in a shoe. Our CGAN would then outputs a set of shoes that reflect these desired features.

For instance, if I wanted an athletic shoe that looked sporty and comfortable, but neither open nor pointy, I would input a vector encoding these preferences and the CGAN would output images of athletic shoes that appear sporty and comfortable, but not open or pointy.

1.6 Narrowing The Scope

While our main ambition is to design the dream shoe generator described earlier, we must first conduct 3 experiments. Our dream shoe generator is only possible with the successful completion of the following:

1. Simple Shoe Generation

   Before creating a CGAN-based shoe generator, we need to be able to develop a regular shoe generator. We will therefore create a regular GAN whose architecture we can later extend to train the CGAN.

2. Functional Type Classification

   To make our shoe generator effective, we need to draw upon as many shoe images as possible. This means using shoe images outside of our dataset. While the images in our dataset already have functional type labels, we need a functional type classifier so that we can add more labeled images to the training set of our CGAN.

3. Attribute Classification

   Similarly, we need a way of assigning attribute labels (open, pointy, sporty, comfortable) to all the images in our dataset so that they can be used to train our CGAN.

In this paper, we will not focus on creating the CGAN but rather on achieving these 3 goals. This paper will act as a stepping stone for the shoe generator of the future.

2 Dataset

The dataset we will be using is the UT-Zap50K dataset [34], which is a dataset consisting of $50,025$ images collected from Zappos.com, the online retailer. The images were curated by researchers at the University of Texas. The images are categorized into 4 major categories - 'shoes', 'sandals', 'slippers' and 'boots'. Within these categories, the shoes are further divided into 21 functional types. For example, the functional type 'oxfords' exists within the 'shoe' category.

Similarly to Khosla and Venkataraman [18], we found multiple issues with the UT-Zap50k dataset. The most glaring issue that we encountered was the lack of uniform image dimensions. We overcame this by finding the image with the smallest dimensions (102×135) and cropping every image to match these dimensions. Because the dimensions of the larger images only varied by 1 pixel at most and the background of each image was white, cropping did not result in the loss of important information. We also found that some of the 21 categories were poorly curated and far too small for our purposes. For instance, the 'boot' functional type contained 13 images of miscellaneous shoe styles. Another class that we removed was the 'prewalker' functional type, which consisted of shoes for infants that have not started walking. The problem with this category was that it was an assortment of all other functional types, just for children. We believed that this would confuse our classifiers. Overall, we decided to remove 10 categories from the initial 21, either because they had less than $1,000$ images or because we believed their content was not well curated. In the end we cut down the data set to $48,442$ images spread across 11 categories. Hence, we retained most of the data ($50,025$ images initially) while significantly cutting down on the number of classes.

These are the 11 categories we were left with are: 'boots-ankle', 'boots-kneehigh', 'boots-midcalf', 'sandals-clogsmules', 'sandals-flats', 'shoes-athletic', 'shoes-flats', 'shoes-heels', 'shoes-loafers', 'shoes-oxfords', 'slippers-flats'.

In addition, we split the dataset into a training, validation and test set with an 8:1:1 split, respectively. We made sure that for each class, a random 10% was in the validation set, another random 10% was in the test set and a random 80% was in the training set.

The UT-Zap50K dataset also offered pair-wise attribute comparisons between shoes. Each comparison contains 2 shoes, A and B, and tells us whether one shoe has more of an attribute than another (figure 1). The 4 attributes compared are 'open', 'pointy', 'sporty' and 'comfort'. The data set contains $11,085$ such comparisons.

Figure 1: Pairwise comparisons of shoe attributes

3 Shoe GAN

Since our end goal is to create a CGAN to generate customized shoe designs, we thought a reasonable first step would be to create a regular GAN for shoes.

Training a GAN on all the images in the UT-Zap50K dataset would very computationally expensive, so we decided to only train the GAN on athletic shoes, which is the largest of the 11 functional types in our dataset.

3.1 MNIST Shoe GAN

3.1.1 Architecture

Our first approach to creating a Shoe GAN, was to model our discriminator and generator after the code we used in assignment 3 to generate images from the MNIST data.

Discriminator: The discriminator has 2 convolutional layers with a leaky ReLU activation, and 2 fully connected layers with a final tanh activation.

Generator: The generator starts with 2 fully connected layers, and passes through 2 transpose convolution layers with a final tanh activation.

Since our training images are much bigger than the MNIST images (102×135 pixels vs. 28×28 pixels) and it is difficult for GANs to converge when trained on large images [12], we decided to shrink our training images to make them more suitable for the GAN's architecture.

3.1.2 Results

During the first few iterations, the generated images seemed promising since each image displayed a clear outline of a shoe (figure 2, top and bottom left). However with more iterations, we found that either the discriminator or generator loss would fall to zero and the quality of the images would deteriorate (figure 2, top and bottom right).

Figure 2: MNIST Shoe GAN Results

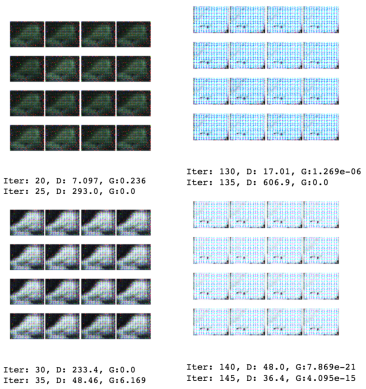


Figure 3: First Training Epoch

Figure 4: Last Training Epoch

3.2 Another Approach: DiscoGAN

3.2.1 Architecture

After our attempt to re-purpose an MNIST GAN for our shoe dataset, we decided to use an existing GAN architecture for large, colored images. In particular, we used the DiscoGAN architecture for the discriminator and generator [19] [6].

Discriminator: The discriminator has a convolutional layer with a leaky ReLU activation, and three sets of convolutional, batch normalization, and leaky ReLU activation layers. This then goes through a fully connected network with a final sigmoid activation.

Generator: The generator starts with a fully connected layer with batch normalization and a leaky reLU activation. There are three sets of a transpose convolution layer, a batch normalization layer, a leaky reLU activation and dropout with $p = 0.5$ after this. This is then passed through another transpose convolution layer with a final tanh activation.

3.2.2 Results

After 1 epoch, we found that the discriminator and generator loss seemed more reasonable than our previous attempt (i.e. nothing in the order of 10^3 or 10^{-3}). The images during the first epoch of training seemed quite reasonable as well (figure 3).

After 15 epochs, the pictures generated by the generator became quite clear and closely resembled actual pictures from the dataset (figure 4).

In addition, at test time, the images looked remarkably like real athletic shoes (figure 5).

3.2.3 Tuning Hyper-parameters

After the promising results, we sought to adjust the architecture of the GAN to make the quality of the images even crisper. We tuned our hyperparamters based on Soumith's tips for training a GAN [5]. In particular, we tried drawing from more noise i.e. $Z \sim Uni(-1, 1)$ rather than $Z \sim Uni(-0.5, 0.5)$, drawing noise from a Gaussian distribution rather than a uniform distribution, using stochastic gradient descent rather than Adam to update the weights, and lastly using a leaky ReLU activation in the final layer of the generator. The test examples for each of these changes are shown in figures 6, 7, and 8 respectively. We believe that the best images are generated when all of the above changes were made (figure 8).

Figure 5: Test Time

3.2.4 Analysis

It seems quite clear from the pictures that the network is stable and generates very realistic images. We found that the biggest increase in image quality was caused by sampling the noise from a range of -1 to 1 rather than -0.5 and 0.5. The other changes did not significantly increase the quality of images when applied on their own. One issue we faced was the lack of an objective metric to measure how athletic the generated images are test-time. Thus, our judgments about the ”best” hyper-parameters are quite subjective.

An interesting pattern we see in these shoes is that random white noise on the sides of the shoes tends to either stretch into the Adidas stripes or some other logo. This makes sense because the Adidas subset in the athletic shoes database is very large compared to the others. We can also see the Vans and Asics stripes.

4 Functional Type Classification

As mentioned previously, we need to create a classifier that accurately labels shoes according to their appropriate functional type so that we can add to the CGAN's dataset in the future. Because our dataset was full of shoes of a similar size, facing the same direction and against the same solid white backdrop, we felt that only a validation accuracy greater than 80% would be acceptable for our classifier.

Figure 6: Noise Sampled from Uni(-1,1)

Figure 7: Uni(-1,1) and Normal Distribution

Figure 8: Uni(-1,1), Normal Distribution, SGD and Leaky ReLU

Figure 9: Simple Classifier Training Results

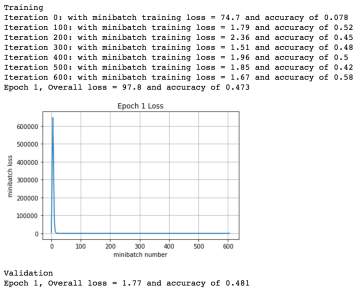
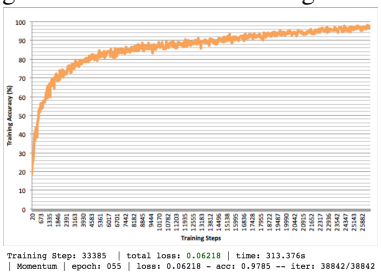


Figure 10: ShoeLeNet Training Results



4.1 Methodology: A Simple Classifier

We first decided to create a very simple classifier. Our simple convolutional network consisted of one convolutional layer followed by a ReLU layer followed by an affine layer and then a softmax cross-entropy loss. Our convolutional layer had a filter size of 7x7, 32 filters, a stride of 1 and used no padding. We experimented with the hinge-loss function first but found that while the training loss decreased monotonically during training, this did not correspond to increases in training or validation accuracy. However, when we switched to softmax-cross entropy loss, we found that in general decreases in the training loss lead to increased training and validation accuracy. In addition, we used batch sizes of 64 images and no regularization.

4.2 Results: A Simple Classifier

Our simple convolutional network was able to achieve a validation accuracy of 48.1% (figure 9). However, after the first 100 minibatches we did not see anymore improvement. We felt like this was a good baseline that allowed us to move on to more complex classification models.

4.3 Methodology: ShoeLeNet

While the results from our simple implementation were much better than random guessing, we were convinced that a more sophisticated architecture could yield even better results. Specifically, GoogLeNet [32] has been run successfully on images that are 224x224 while remaining computationally efficient. The key insight is that a bundle of smaller convolutional layers can produce the same result as one larger, more inefficient layer. We modified the GoogLeNet architecture [8] by changing the first layer and the last activation layer to match the dimensions of our images (102x135).

4.4 Results: ShoeLeNet

The results we garnered were very good. Over 55 training epochs, we were able to achieve 97% training accuracy (figure 10). This was not solely over-fitting as we also measured a validation accuracy of 88%. Since we achieved our target for the functional type classifier, we moved on to creating the attribute classifier.

5 Binary Classifier For Shoe Attributes

5.1 Motivation

As mentioned in the dataset section, in addition to shoe images, the UT-Zap50K dataset also includes $11,085$ pair-wise comparisons between shoes. These comparisons compare four specific attributes of shoes. These attributes are openness, pointedness, sportiness, and comfortableness. Examples of these comparisons are shown in Figure 1.

Note that each pair-wise comparison in the dataset only compares shoes based on one attribute. Since the overarching goal of the paper is to lay the foundations for a conditional generative adversarial network (cGAN), we need a way to classify all $50,000$ images in the dataset as open or not-open, pointy or not-pointy, sporty or non-sporty and comfortable or not-comfortable. Note that

these attributes are not mutually-exclusive. In section 4 we demonstrated that it is possible to create a generative adversarial network (GAN) for athletic shoes. In this section, we attempt to use the pairwise comparisons in our dataset to train a convolutional neural net to determine whether a shoe is pointy or not pointy (binary classification). If such binary classification is possible, then this technique can be extended to train binary classifiers for the three other attributes as well. Once we have all four binary classifiers, we could then run each of the $50,000$ shoes through each classifier to determine which attributes they posses.

5.2 Methodology

The first step in training our binary classifier was to create a training set from the comparison data. We did this by, rather crudely, taking each comparison and giving the less pointy shoe a label of 0 (non-pointy) and the more pointy shoe a label of 1 (pointy).

Of the original $11,085$ comparisons in the dataset, $2,700$ compare the pointedness of shoes. In addition for each comparison, the 5 amazon turkers that created the comparisons were required to specify how confident they were about the ordering of the comparison. These confidence scores were on a scale of 1 to 3 (1 being very confident and 3 being not very confident). Each comparison also featured the fraction of turkers who gave the majority vote (1.0 meaning all turkers agreed on the directionality of the comparison).

In our preliminary experiments, we only wanted to train on very strong orderings and therefore only chose comparison examples with an average confidence score of 1.0 (i.e. all 5 turkers were very confident in their decision) and with 100% turkers giving the majority vote.

After all this pruning, we were left with 730 pointy comparisons. Each comparison contains two images and resulting in $1,460$ images. We decided to use 90% of these images for our training set, leaving 5% for the validation set and 5% for the test set. Overall, we had $1,314$ training examples, 73 validation examples, and 73 test examples.

Figure 11: Resizing Images For VGG Network

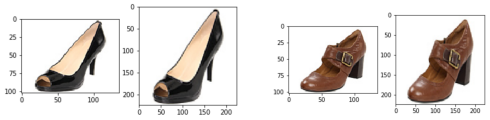
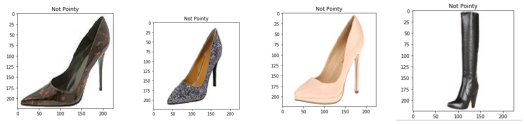


Figure 12: Incorrectly Classified as 'Not Pointy'



5.3 VGG Network For Binary Attribute Classification

We decided to use TFLearn's VGG Network for the Oxford Flowers 17 classification task but train the weights from scratch [27] [**vgg-simonyan2014very**]. We modified the architecture slightly so that the last fully connected layer has 2 units instead of 17. Since the VGG Network takes in images with dimensions $224 \times 224 \times 3$ we had to resize our images from $102 \times 135 \times 3$. We do not believe that this resizing significantly distorted our images as we can see from figure 11.

5.3.1 Training VGG Network

We trained the VGG Network on the 1314 training images for 50 epochs using an RMSProp optimizer with a learning rate of 0.0001.

5.3.2 VGG Network Results

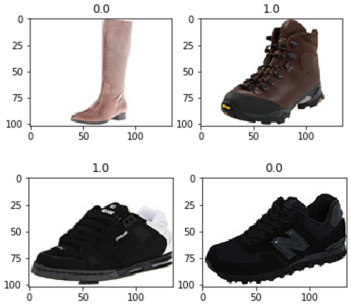
The results from the VGG Network are not promising. The final training accuracy was 54.03% and the final validation accuracy was 46.67%. In addition, we found that the network predicted almost all of the images in our validation set as non-pointy (figure 12), which suggests that the network was unable to learn features in the input images that correspond with pointedness.

5.3.3 Hypothesis About VGG Results

We believe that the reason why the VGG network was unsuccessful was down to our method of labeling shoes

Figure 13: Labeling Images With Comparison Data

Figure 14: Incorrect Comparisons



as pointy and non-pointy based on the directionality of comparisons. For example, if a comparison contains two shoes that both aren't pointy compared to the rest of the dataset, then labeling one of those shoes as pointy may have confused the network and hindered its learning. Likewise, if both images in a particular comparison are very pointy (relative to the rest of the dataset) and we labeled one of the images as not pointy, this also could have confused the network. An example of this problem is illustrated in figure 13, where both shoes are pointy and yet we labeled the right image non-pointy (class 0).

Furthermore, some comparisons in the dataset were seemingly incorrect (fiqure 14) which is another factor that may have hindered the VGG network's learning.

5.4 Inputting Both Images Simultaneously

It is worth mentioning other approaches that we used to overcome the issue caused by labeling shoes as pointy or non-pointy based on the directionality of the comparison. One alternative approach that we explored was to create a CNN that accepts both images from the same comparison as inputs.

The architecture of this CNN was a convolution layer C_1, a max-pooling layer P_1, a convolutional layer C_2, a max-pooling layer P_2, and then a fully connected layer. C_1 had 32 filters of size 5 by 5, no padding, a stride of 1, and a ReLU activation. C_2 had 64 filters of size 5 by 5, no padding, stride of 1, and a ReLU activation. P_1 and P_2 both had a window size of 2 by 2 and a stride of 2. The fully connected layer had a single unit (the pointy score for the input image).

Since a single comparison consists of two images, A and B, both images are fed into the CNN resulting in two scalars, a and b, representing the raw scores for each image's pointedness. We then concatenated a and b to produce the row-vector s. Next we applied a softmax to s to convert the row-vector into an estimated probability distribution, p. Lastly, we used a cross entropy loss, where the "true" probability distribution is given by q, where $q = [1, 0]$ if shoe A is pointier than shoe B. Otherwise, $q = [0, 1]$.

We also experimented with a different architecture that used a hinge-loss rather than a softmax cross-entropy loss. Specifically, the concatenation of raw scores for Shoe A and Shoe B, s, was passed through a sigmoid layer to produce p. If shoe A was pointier than shoe B, then $L_i = max(0, p_B - p_A + \delta)$ otherwise $L_i = max(0, p_A - p_B + \delta)$. We used $\delta = 1$ to enforce as large a margin as possible between p_B and p_A.

5.4.1 Training

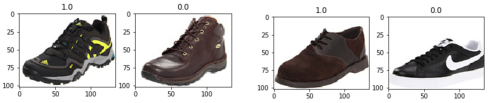
For both architectures (softmax cross-entropy loss and hinge-loss), we trained the model using stochastic gradient descent with a batch size of 64. In addition, we used an Adam Optimizer with a learning rate of 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e - 08$.

5.4.2 Results

The results for both architectures were not promising. In the softmax variant, across multiple runs, the average loss across a batch consistently flat-lined at 0.693, which is also $-log(0.5)$. This means that the network is giving equivalent probabilities and raw scores to both shoe A and shoe B i.e. $a = b$. Therefore the network is not learning the features that correlate with a shoe's pointedness and just ends up guessing which shoe is more pointy.

In the hinge-loss variant, across multiple runs, the average loss across a batch flat-lines at 1, which is the value of δ that we used. Therefore we have $p_A = p_B$, i.e. the

Figure 15: Ambiguous Comparisons



raw scores for shoe A and shoe B are equal. Once again the network is not learning what features to focus on to determine a shoe's pointedness.

5.4.3 Hypothesis About Results

As was the case with the VGG Network, we believe that the inability of our alternative model to learn was due to shortcomings in the comparison dataset. Specifically, many of the comparisons in the dataset were simply just too close to call, which may have confused the network. If shoe A and shoe B seem equally pointy and yet A is labeled as more pointy, the network may get confused as to what features in the input image to focus on. We found the large number of ambiguous comparisons to be very surprising especially given that we only selected comparisons with an average confidence score of 1.0 and agreement among turkers of 100%. The images in figure 15 illustrate some comparisons that were "too close to call".

6 Future Considerations

It is evident that one of our biggest obstacles for creating the CGAN is the dataset, specifically the attribute comparison part of the dataset. In particular, the attribute comparisons provided seem unjustified, vague and very subjective. Future approaches may consider finding a different dataset (e.g. DeepFashion [23]) to classify attributes.

Once we build an effective attribute classifier, we will have all the ingredients to train a conditional GAN of the future!

References

[1] Agnés Borras et al. "High-level clothes description based on colour-texture and structural features". In: *Pattern Recognition and Image Analysis* (2003), pp. 108–116.

[2] Peter Burt and Edward Adelson. "The Laplacian pyramid as a compact image code". In: *IEEE Transactions on communications* 31.4 (1983), pp. 532–540.

[3] Ju-Chin Chen and Chao-Feng Liu. "Deep net architectures for visual-based clothing image recognition on large database". In: *Soft Computing* (2017), pp. 1–17.

[4] Ju-Chin Chen and Chao-Feng Liu. "Visual-based deep learning for clothing from large database". In: *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM. 2015, p. 42.

[5] Soumith Chintala. *How to Train a GAN?* `https://github.com/soumith/ganhacks`. 2016.

[6] Gunho Choi. *Tensorflow Implementation of DiscoGAN*. `https://github.com/GunhoChoi/DiscoGAN_TF`. 2016.

[7] Emily L Denton, Soumith Chintala, Rob Fergus, et al. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks". In: *Advances in neural information processing systems*. 2015, pp. 1486–1494.

[8] Burness Duan. *Google Net*. `https://github.com/tflearn/tflearn/blob/master/examples/images/googlenet.py`. 2016.

[9] *Footwear Market - Global Industry Analysis, Size, Share, Growth, Trends, and Forecast 2015 - 2023*. 2015.

[10] Jon Gauthier. "Conditional generative adversarial nets for convolutional face generation". In: *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester* 2014.5 (2014), p. 2.

[11] Ross Girshick et al. “Region-based convolutional networks for accurate object detection and segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 142–158.

[12] Ian Goodfellow. *Do generative adversarial networks always converge?* 2016. URL: https://www.quora.com/Do-generative-adversarial-networks-always-converge.

[13] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[14] Kota Hara, Vignesh Jagadeesh, and Robinson Piramuthu. “Fashion apparel detection: the role of deep convolutional neural network and pose-dependent priors”. In: *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE. 2016, pp. 1–9.

[15] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[16] Geremy Heitz and Daphne Koller. “Learning spatial context: Using stuff to find things”. In: *Computer Vision–ECCV 2008* (2008), pp. 30–43.

[17] Junshi Huang et al. “Cross-domain image retrieval with a dual attribute-aware ranking network”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1062–1070.

[18] Neal Khosla and Vignesh Venkataraman. “Building image-based shoe search using convolutional neural networks”. In: *CS231n course project reports* (2015).

[19] Taeksoo Kim et al. “Learning to discover cross-domain relations with generative adversarial networks”. In: *arXiv preprint arXiv:1703.05192* (2017).

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[21] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[22] Christian Ledig et al. “Photo-realistic single image super-resolution using a generative adversarial network”. In: *arXiv preprint arXiv:1609.04802* (2016).

[23] Ziwei Liu et al. “DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[24] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).

[25] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks”. In: *arXiv preprint arXiv:1601.06759* (2016).

[26] Aaron van den Oord et al. “Conditional Image Generation with PixelCNN Decoders”. In: *arXiv preprint arXiv:1606.05328* (2016).

[27] Andy Port. *VGG Network*. https://github.com/tflearn/tflearn/blob/master/examples/images/vgg_network.py. 2015.

[28] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).

[29] Scott Reed et al. “Generative adversarial text to image synthesis”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 3. 2016.

[30] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[31] Yi Sun, Xiaogang Wang, and Xiaoou Tang. “Deeply learned face representations are sparse, selective, and robust”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2892–2900.

[32] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[33] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. “Generating videos with scene dynamics”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 613–621.

[34] A. Yu and K. Grauman. “Fine-Grained Visual Comparisons with Local Learning”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2014.

[35] Jun-Yan Zhu et al. “Generative visual manipulation on the natural image manifold”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 597–613.