

Building Image-Based Shoe Search Using Convolutional Neural Networks

Neal Khosla
Stanford University
nealk@cs.stanford.edu

Vignesh Venkataraman
Stanford University
viggy@stanford.edu

Abstract

In this paper, we explore the applications of convolutional neural networks towards solving classification and retrieval problems as they pertain to images of shoes. Having collected a dataset of over 30,000 shoes, we attempt to classify each shoe into its appropriate shoe category as well as to retrieve the top 5 most similar shoes in the dataset. To solve each of these problems, we experimented with multiple network architectures. For the problem of classification, we determined that even a shallow 3 hidden layer network was able to classify the shoes with accuracies above 90%. For retrieval, we eventually settled on using transfer learning to apply the VGGNet architecture to our problem. Extracting the last fully connected layer from a pretrained model gave us feature vectors for each shoe image in the dataset. Computing Euclidean distances between these feature vectors and that of our query image and returning the closest matches allowed us to achieve 75.6% precision on retrieval and an average subjective quality score of 4.12/5. On the whole, convolutional neural networks allowed us to classify and compare shoes at a remarkably high degree of accuracy, significantly outperforming prior work done on this specific problem.

1. Introduction

Since the term artificial intelligence was coined, mankind has sought to train computers to see as we do. One of the most overlooked aspects of human cognition is sight, and when we consider the scope of the problem from a computing perspective, we realize just how powerful the human brain actually is, going from the real world to a series of firing neurons to recognition and response in (literally) the blink of an eye.

There is immense research being done in the field of visual search, based loosely around the adage that a picture is worth a thousand words. On the internet today, however, the converse may in fact be true: it is easy and effective to do a Google search with plain text, but doing a visual search with an uploaded image yields unpredictable results. With

picture data being readily available, and an ever-increasing number of mobile devices being given cameras and connections to the internet, we predict that interest in this field will continue to grow, bringing with it a desire for services that accept pictures as input and return recommendations and insights based on them. This project seeks to take a first step towards achieving that goal.

Specifically, we have chosen to focus on shoe recommendations and comparisons as our service of choice. Shoes are a popular consumer product, with sufficient variety in market for a service built around them to be useful. Shoes are also almost entirely bought and sold on visual appeal - the appearance of a shoe is generally what people gravitate towards, and when shoes are vocally appreciated or admired, the conversation often trends towards where they were bought and how much they cost. Moreover, shoes are mainly differentiated on three visual characteristics - shape, texture, and color [12]. Convolutional neural networks are renowned for their ability to learn basic shapes, textures, and colors, making this problem a good fit for the application of neural networks [8].

Overall, our approach is to train one or more convolutional neural networks on large shoe datasets, and then use these trained networks to perform two tasks: classify input images into shoe classes, and perform a nearest-neighbors analysis to return the shoes most similar and most relevant to the input.

2. Background and Related Work

Given the inefficiencies in the retail space, this is a problem that many people are trying to tackle in the real world right now. A clothing recommendation engine, Kaleidoscope, initially tried to recommend similar clothing items to its users by utilizing machine learning-based classification methods but ended up settling for manual classifications due to an inability to properly classify the stylistic elements of retail goods that they were assessing. Another company that was actually fairly successful at classifying similar retail goods on the basis of images was Modista [10], a computer vision and machine learning based retail good recommendation service. We spoke with one of the co-founders of

Modista to gather his perspective and he mentioned that his team examined similarity between shoes primarily on the basis of shape and color, which is evident in the image below that is an example recommendation from their website. The Modista website also requires users to click through multiple images to generate a similar image and is an interface that is high friction and painful for users.

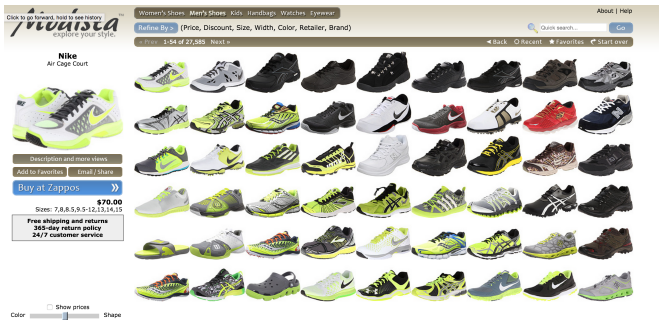


Figure 1. A set of shoe recommendations generated by Modista.

There has also been some non-commercial, academic work in this space. While progress has been made to solve individual subproblems of this problem, none of the work has been particularly successful at creating a very accurate and robust image-based recommendation engine.

We also examined work by Borrás et al. through a UAB Bellaterra paper entitled “High-Level Clothes Description Based on Colour-Texture and Structural Features” [1]. In this work, the authors seek to interpret five cloth combinations in a graphical structure to try to ascertain what someone is wearing from an image. While the work that the researchers did is interesting, they only achieved a 64% overall accuracy. This work demonstrated the difficulty of this problem in a natural setting. In addition, this paper represents another example of work that is using traditional computer vision and machine learning techniques to attack similar problems, but does not use the deep learning techniques explored by this paper.

Although it is also not exactly the same problem, the work done by Bossard et al. [2] is another example of academic attempts to solve problems related to the fashion and retail industry using core computer vision principles. This paper again demonstrates the shortcomings of traditional techniques in solving problems of reasonable complexity.

Lastly, we cite work that we have previously done towards this project as work for previous classes we have taken, CS 229 [14] and CS 231A [15]. In both attempts, we used traditional methods in Machine Learning and Computer Vision accordingly, to attempt to solve this problem. It was evidently clear that such methodologies were not robust enough to perform this task at a very high level, which was part of what drove us towards solving this problem using deep learning techniques. We topped out at roughly 60%

classification accuracy, even with very generic labels, and could never get more than half of the recommendations correct for the retrieval task.

3. Approach

Our problem is clearly twofold, and consists of both classification and retrieval - we seek to take an input image, figure out what kind of shoe it corresponds to, and also return the closest visual matches to it, essentially serving as a computerized shoe-store employee. We present below the various approaches considered for each of the tasks in turn:

3.1. Classification

This part of the problem is a canonical, textbook usage of convolutional neural networks. Our first approach was to pick a dataset and train a network from scratch, tuning parameters to maximize efficacy and accuracy. This approach would require a moderately sized dataset and significant computational resources, proportional to the depth of the network chosen [4]. This approach is also the most obvious application of neural networks - for a straightforward classification problem, the general model is to train a classifier and tune those results [5]. For the sake of completeness, especially given the fact that we had no idea whether this approach would perform well at all, we formulated other courses of action to be applied in the event of poor results. Our first backup approach was to rely on fine-tuning another network using our selected shoe dataset in an application of transfer learning [7]. This approach would require (again) significant computational resources, but we would be able to use a smaller dataset, since much of the training process would have already occurred while finalizing the weights of the original network we are fine-tuning. In the event neither of these approaches worked, we were willing to reduce the number of labels and make the class distinctions more obvious - for instance, removing the distinction between loafers and oxfords. We expected that this would boost network accuracy by making the classification problem simpler.

Evaluating the performance of the classification problem is easy - using a withheld validation set, we would be able to see how accurate our network ended up being and use that accuracy metric to gauge how well the network is doing [6]. In general, the higher the (training and validation) accuracy, the better. We also hoped to test our network against images not in the official dataset, and ideally from the real world (i.e. our own cameras); this would be a powerful indicator as to the adaptability of our network to a wider range of input data.

3.2. Retrieval

The retrieval aspect of the problem is a less canonical usage of convolutional neural networks. The key to this part is boiling each image down to a lower-dimensional

representation X_i , then comparing an input image’s lower-dimensional representation X' to each X_i using some distance metric, returning the “closest” matches to the input in a manner akin to nearest neighbor analyses [11]. The trick, then, is to pick a good lower-dimensional representation and a good distance metric. We readily picked L2 distance as our metric of choice, both for its computational simplicity, its roots in Euclidean geometry, its popularity, and our familiarity with what it means in a literal sense:

$$L2(i, j) = \sqrt{(X_i - X_j)^2}$$

As seen above, the L2 distance between images i and j is the 2-norm of the difference between their chosen representations X_i and X_j . Having picked our distance metric, our first approach for this part was to use the trained-from-scratch network from the classification task as a feature extractor rather than a classifier - without any knowledge of the actual network architecture, treating the last-but-one fully connected layer as a feature vector for an image may yield good results, especially if the network itself performs well in the classification task [7]. After training the network, this approach would be computationally simple and just require a single pass back over our dataset to extract features for each image. After these features are stored, all that remains is to extract the features for a given input image and perform a one-to-many comparison over the stored features, returning the images that have the smallest L2 distance separating them and the input image. As in the classification task, we were unsure as to whether this approach would work and thus made some alternate plans as well. Our first backup approach was to again rely on transfer learning - rather than use our own network as the feature extractor, we would use another pretrained network, either with or without fine tuning, as the feature extractor, again taking the second-to-last fully connected layer as the feature vector output for each image [7]. The rest of the process remained the same. This strategy would allow us to test drive a variety of networks in the feature extractor role and pick the one that worked the best.

Evaluating the performance of a retrieval problem is a much harder task as compared to the classification problem, essentially because the problem statement itself is fuzzier! In the classification problem, there is absolutely a correct answer, in that a running shoe can only ever be a running shoe and a boot can only ever be a boot. The retrieval problem wants our engine to recommend shoes that “look similar,” which is an inherently subjective, rather than objective, criterion. Here, “good” results are visually similar images, and “bad” results are dissimilar images. We planned to either randomly sample enough of these results to get a statistically significant confidence interval of accuracy, or feed the results through mechanical turk to have real humans evaluate our success [14, 15].

4. Experiments

Having defined our approach in the section above, we had three main areas of experimentation - our dataset, our classification networks, and our retrieval networks. We also experimented with various computational resources, ranging from local machines to Terminal.com instances. We will detail our full development process in the following subsections.

4.1. Computational Resources

We had initially hoped to develop our entire project locally, as we had done in previous artificial intelligence classes. However, convolutional neural networks are on another scale entirely when it comes to memory and CPU/GPU usage, and as such we had to look for other options. Even on our Apple MacBook Pros, with 16 GB of RAM and nVidia GT 650M/750M graphics cards, we were unable to efficiently train even the shallowest convolutional neural networks without butchering our ability to get any other work done. Another factor that pushed us towards cloud computing was the difficulty installing Caffe [3], the framework-of-choice for this class. When the teaching staff offered us the option of preconfigured Terminal.com GPU instances, complete with installed Caffe and ssh access, we jumped at the opportunity. In retrospect, this may not have been the ideal decision, as Terminal.com proved to be spotty and unreliable for much of the development process. We were unable to ever actually ssh into a Terminal.com instance, and it seemed there were never any GPU instances available whenever we sat down to work on the project. Their billing practices also seemed quite strange, as we’d work for just about 15 minutes, pause the instance, and return to find two dollars missing from our account credits. On the whole, if given the opportunity to select again, we heard favorable things about GPU-configured AWS instances, with some groups claiming that they were able to find relatively recent Caffe snapshot images ready for deployment through EC2. However, we were able to extract good results from Terminal.com when it was working, and thus our complaints are mostly irrelevant as they pertain to the success of our project.

4.2. Dataset

Initially, we hoped to use the UT-Zap50K dataset [16], a 260MB archive culled from Zappos.com by a research group at the University of Texas, as our primary knowledge base, as it appeared to be a labelled and pruned dataset that satisfied our needs. It purportedly contained 50,000 unique shoes that were organized via a topologically sorted list of categories, as determined by a combination of the research group and workers from mechanical turk. The images were all supposed to be uniformly sized and compressed with minimal artifacting, and the class labels were supposed to

have been vetted with over 95% accuracy. However, rough testing and visual inspection of this dataset made it very clear to us that it was unusable. Specifically, many of the images were miscategorized, the class labels themselves were unspecific and vague, the images were severely compressed (with tons of artifacts) to an odd aspect ratio, many images were of a different size than the stated default, the images were taken at an odd angle (from the front and above as opposed to a sideways shot), there was a preponderance of shoe duplication, there were far too many children's shoes, and there was no way to trace a shoe image back to its origin on Zappos.com. These limitations made it impossible for us to have much success testing either the classification or retrieval aspects of our task. As such, we had to look at other dataset options.

Since we were so enamored with the cleanliness and the uniformity of shoe images on Zappos.com, we chose to stick with the site as our source of images. Instead of using the UT-Zap50K dataset, we decided to scrape our own manually. This process involved dissecting the URL structure of Zappos.com and a fairly extravagant HTML tag-parsing exercise, but after much trial and error, we were able to write a script that would successfully go through and scrape an entire gender's worth of shoes from Zappos. The choice of scoping by gender was made to limit the runtime of the script - going for all the shoes listed on Zappos in a single run would take about 8 hours, whereas scoping by gender would allow for 3.5 hour to 4.5 hour parallel scraping runs. After some delay and some internet connection snafus, we were able to assemble a dataset of nearly 12,000 men's shoes and 20,000 women's shoes. Each shoe would have two key elements tied to it: a shoe image taken from the side (the image data) and a small text file with shoe name, brand, price, SKU, rating, color, and category (corresponding label(s)). An example of this is shown below:



Figure 2. Shoe image number 33 in the scraped dataset.

```
33
Nike Roshe Run Challenge
Nike
Red/Laser Crimson/Midnight Navy/White
Shoes > Sneakers & Athletic Shoes > Nike
0
```

```
$75.00
5
SKU 7925931
Corresponding metadata text file for shoe number 33.
```

Each image had exact dimensions 1920x1440 (high-resolution 4:3 aspect ratio), with file sizes ranging from 100 KB to 600 KB. In total, the men's shoes dataset took up about 4 GB of storage uncompressed, while the women's shoes dataset used up about 6 GB of storage uncompressed.

We had initially hoped to send our data through mechanical turk and have turk workers label each shoe with more refined category information, but a combination of cost, difficulty setting up the turk task, and some egregiously bad customer support from Amazon Payments made the mechanical turk dream impossible given the time constraints imposed by the project. However, in retrospect, this may have worked out in our favor, as many fellow students expressed dissatisfaction with how well turk workers managed to tag their data. In contrast, Zappos.com simply doesn't have mislabeled or miscategorized shoes, and their labels appear specific enough to get the job done. Some examples of shoe labels include "Athletic," "Loafers," "Sandals," "Boat Shoes," and "Boots." These categories are different enough for there to be obvious visual differences between classes, but homogeneous enough within their own contexts so as to make recognition (theoretically) simple. In total, there were twelve shoe classes in this dataset.

A final challenge we faced with our dataset was moving it around - extracting the dataset required a programmatically accessible browser in Selenium and Google Chrome, and thus we had to run the scrapers locally. However, shunting 10 GB of data into the cloud and/or onto a Terminal.com instance proved debilitating, as for whatever reason three different Terminal.com instances refused to accept our ssh keys as valid. As a hacky workaround, we uploaded our dataset to AFS and then copied it from there onto Terminal.com, which took much longer but ended up working out perfectly.

4.3. Network Architectures

In conceptualizing this project, we hoped that we would get the chance to try out a variety of neural network architectures on both the classification and retrieval problems. The use of Caffe [3] as our neural network framework made this a very simple task, as the only work needed was to modify a protocol buffer file with architectural changes. We experimented briefly with a number of architectures, but performed the vast majority of our trials and our training on the following three networks:

4.3.1 ViggNet Small

This small three layer network was intended to be our sanity check network, suited for the task of making sure our code actually worked. Its small size made training and testing very quick, and it didn't have the heinously large memory footprint of the larger networks. However, it was also remarkably performant, as later results will show. The network architecture was as follows:

$$INPUT \rightarrow [CONV \rightarrow RELU \rightarrow POOL] \times 2 \\ \rightarrow FC \rightarrow SOFTMAX$$

This setup had depth 32 at each convolutional layer, using standard 3x3 filters with pad 1 and stride 2 (thus preserving the input size through convolutions, and halving the image size after each size 2 stride 2 pooling layer) [4]. The FC layer had 12 outputs, matching the number of classes in our classification problem, and mapped directly to a softmax loss layer and an accuracy layer.

4.3.2 ViggNet Large

This larger multi-layer network was the network that we felt had the best chance for success. Based off the VGGNet architecture [13], it was just a much smaller version of the original behemoth; this reduction was made in the interest of memory usage and training time until convergence. Its architecture was as follows:

$$INPUT \rightarrow [CONV \rightarrow RELU \rightarrow CONV \rightarrow RELU \\ \rightarrow POOL] \times 2 \rightarrow FC \rightarrow RELU \rightarrow DROPOUT \\ \rightarrow FC \rightarrow SOFTMAX$$

This setup had depth 64 in the first two layers, depth 128 in the second two layers, depth 512 in the first fully connected layer, and 12 outputs right before the softmax corresponding to 12 shoe classes. Again, we used standard 3x3 filters with pad 1 and stride 2, preserving input size through convolutions and halving dimensions with each pooling layer [4, 13]. Because of the memory requirements of this network, we had to reduce the batch size for training and testing down to 10 images at a time.

4.3.3 VGGNet

This network, borrowed directly from the Caffe Model Zoo [3], was the full VGGNet [13] network that achieved remarkable performance on the ImageNet challenge. Its architecture was as follows:

$$INPUT \rightarrow [CONV \rightarrow RELU \rightarrow CONV \rightarrow RELU \\ \rightarrow POOL] \times 5 \rightarrow [FC \rightarrow RELU \rightarrow DROPOUT] \times 2 \\ \rightarrow FC \rightarrow SOFTMAX$$

This setup had exponentially increasing depth in each layer, culminating in a first fully connected layer with 4096 outputs. It again used the canonical 3x3 filters with pad 1 and stride 2, and max pooling was done with 2x2 filters and stride 2 [4, 13]. The memory requirements of this network proved too much to actually train, and thus it was used primarily in a transfer learning sense (though it was not fine-tuned because of the same memory constraints) with regards to the retrieval problem.

4.4. Classification

We now present our experimental method and results for each of the convolutional neural network architectures shown above:

4.4.1 ViggNet Small

The first network we ran, ViggNet Small, was a three hidden layer network intended to give a quick and dirty representation and classification of our dataset. Instead, we found that this network worked quite well for classification. In just over two minutes of train time, this network was able to go through 1500 iterations and reach a 92% validation accuracy with a loss of only 0.2217. We did try training this for longer and with different parameters but found that the classification accuracy had pretty much plateaued. We are unsure as to why this network performs so well, especially since it doesn't have very much capacity or depth. We hypothesize that the classification problem is actually less difficult for computers than we make it out to be, and thus even with a shallow architecture the network is able to give accurate classifications. It seems that the distinctions between shoe classes and the similarities within a shoe class are readily visible to the network.

4.4.2 ViggNet Large

The second network we ran, ViggNet Large, was a six hidden layer network which we trained in hopes of approximating the effects of VGGNet itself, while being able to train it given our reduced computational resources. Unlike ViggNet Small, this network did not classify that well, plateauing at an accuracy of 64% and taking significantly longer to train. We suspect that further parameter tuning and/or training for longer would potentially improve the validation accuracy of this network, but given the resounding success of ViggNet Small, we felt that our time would be better spent in other areas.

4.4.3 VGGNet

The final network we used was VGGNet from the Caffe Model Zoo [3]. We wanted to try training this network from scratch and also fine-tuning the pretrained model using our

dataset, but we were unable to do either, owing to the network’s massive memory footprint and enormous overhead. We fundamentally could not start training the network, even with image batches of size 10. If we had better computational resources, we would expect that, with some parameter fine tuning and a lot of time, this model would perform extremely well, especially since the VGGNet model achieves amazing accuracy on the much more difficult ImageNet challenge. However, given the accuracy and speed of Viggynet Small, we didn’t lose much sleep over not being able to properly train VGGNet, instead figuring that we would use this network for our retrieval task. Additionally, as the VGGNet model is known to be notoriously difficult to train [13], we felt that attempting to replicate this training process on a much smaller dataset and a much smaller computer would be futile and pointless.

4.4.4 Summary

The following table summarizes the Classification results:

	Train Time	Iterations	Accuracy	Loss
Small	119 s	1500	92%	0.2217
Large	1296 s	1500	64%	0.5742
VGGNet	N/A	N/A	N/A	N/A

4.5. Retrieval

The other half of our report focused on using these network architectures to solve the retrieval problem - namely, given a query image, to be able to retrieve similar and relevant shoes from the dataset. While the evaluating the performance of these networks in this light is a bit more difficult, we chose to evaluate this using a subjective quality rating as well as a standard precision metric [9]. In particular, 100 shoes were randomly sampled from the dataset and submitted as query images. The top 5 results retrieved were then subjectively rated on a scale from 1-5 in terms of their relevance. Each of the top 5 responses for the query were deemed relevant if they were given a 4 or a 5, and not relevant otherwise - this was done to measure the number of query responses that were deemed as good matches or that might reasonably approximate the type of shoe that might be suggested as similar by a salesperson in a shoe store. The precision was then calculated by the following formula [9]:

$$\text{Precision} = \frac{\# \text{ relevant items retrieved}}{\# \text{ retrieved items}}$$

The top 5 ratings were also averaged to give a general idea of how each network performed in terms of the pure quality of the responses.

4.5.1 Viggynet Small

While the Viggynet Small network showed promising results in terms of classification, it was clear that the downside of such a small network without robust feature dimensionality was the inability for the network to properly differentiate between shoes for the purpose of retrieval. It appears as though the features extracted from this network were not the best metrics by which to judge shoe similarity. In particular, this network demonstrated much lower retrieval metrics, having only 62.6% precision and an average score of 3.64.

4.5.2 Viggynet Large

The Viggynet Large network showed slight improvements over the Viggynet Small network, but even still did not perform all that well in terms of quality of recommendations. The network showed marked improvement in terms of precision, retrieving relevant shoes 69.4% of the time, but very marginal improvements in the average score as this only rose to 3.71. This indicated to us that there were further improvements to be had, particularly with regards to average score.

4.5.3 VGGNet

The VGGNet was the most promising of the three networks in terms of retrieval. In particular, the robust features of the network demonstrated an ability to consistently bring up relevant, high-quality matches for any query shoe. This was reflected in the network’s 75.6% precision metric and average score of 4.12. This represents a 12.6% improvement in average score over Viggynet Small. This performance is even more impressive given that this network was not fine-tuned for our dataset, something we surely would have done given more computational power and time. Below is a query and response example demonstrating a characteristic output of our retrieval engine using VGGNet:

Query Image



Retrieved Responses (top 3)



4.6. Summary

The following table summarizes the retrieval task results:

	Average Score	Precision
ViggyNet Small	3.64	62.6%
ViggyNet Large	3.71	69.4%
VGGNet	4.12	75.6%

Overall, all of our networks performed moderately well on the retrieval tasks, but the VGGNet in particular stole the show with its remarkable average match score and high precision. Retrospectively speaking, we realized that scoping our results by shoe class would definitely improve our performance further - that is, using the ViggyNet Small network for classification and then filtering potential matches by the shoe category would limit the candidate pool to only shoes of the same type, which should theoretically improve our metrics.

5. Conclusion and Future Improvements

The preliminary results of this project demonstrate a significant improvement over our previous attempts to solve this problem using traditional computer vision and machine learning techniques - a testament to the power of deep learning methods. We believe that these results represent a very good starting point for future improvements on this project. It is clear to us that this work validates the hypothesis that this is a solvable problem. We hope to further refine this work and make it available to the masses in some capacity, as well as expand this work into other consumer goods and products.

In particular, we derived a couple of takeaways from this project that will inform our next steps in working on this problem. First off, we have realized the difficulty of fine-grained classification and learning problems and, in particular, how much the quality of the dataset and the accompanying labels impact that. This first became clear to us while using the UT-Zap50K dataset, which had low quality labeling and organization and downsampled images, making it nearly impossible for us to train any semblance of an accurate model. After gathering our own dataset, we were able to move forward, but became limited in some respects by the quality of our labels. Given that nearly 45% of the labels in our dataset are sneakers, the system struggles most at differentiating between these types of shoes as there is no separation between things such as tennis shoes or skateboarding shoes, although they look very different. Thus, we believe that better labels with more granularity would improve the quality of our dataset and therefore the performance of our system.

The improvement of our dataset would also allow us to properly and effectively implement another strategy we

believe would improve the quality of our retrieval system. Specifically, when considering nearest matches for our shoes in a k-nearest neighbor strategy, we would like to first filter by the labeling of the shoes, which would allow us to narrow the possible response for our retrieval and hopefully result in more accurate and relevant responses.

Lastly, we believe that with access to stronger computational resources, we could improve on the best model, the VGGNet, through the use of transfer learning by fine-tuning the weights to our specific dataset. In addition to the data we have aggregated from Zappos, we are in the process of scraping data from many other sites on the web so that we can build a larger dataset and train a more robust system with a larger range of potential responses. It became clear to use through this process that the depth and capacity of the network were vital towards building an accurate retrieval system and so having more computational resources would allow us to match the increasing quality of our dataset with better trained and higher capacity models. We also plan to test our network(s) against photos of shoes taken in real life to see how well they adapt to noisy real-world images.

Overall, we are very pleased with our results and hope to build upon them in the months to come.

References

- [1] Agnes Borràs et al. "High-Level Clothes Description Based on Color-Texture and Structural Features". In: *Lecture Notes in Computer Science*. Vol. 2652. Iberian Conference, Pattern Recognition and Image Analysis (IBPRIA'03). Palma (Spain): Springer-Verlag, June 2003, pp. 108–116. URL: <http://www.cat.uab.cat/Public/Publications/2003/BTL2003>.
- [2] Lukas Bossard et al. "Apparel classification with style". In: *Computer Vision—ACCV 2012*. Springer, 2013, pp. 321–335. URL: http://www.vision.ee.ethz.ch/~lbossard/bossard_accv12_apparel_classification_with_style.pdf.
- [3] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv preprint arXiv:1408.5093* (2014). URL: <http://caffe.berkeleyvision.org>.
- [4] Andrej Karpathy. *Convolutional Neural Networks*. 2015. URL: <http://cs231n.github.io/convolutional-networks/>.
- [5] Andrej Karpathy. *Image Classification*. 2015. URL: <http://cs231n.github.io/classification/>.

- [6] Andrej Karpathy. *Linear Classifiers*. 2015. URL: <http://cs231n.github.io/linear-classify/>.
- [7] Andrej Karpathy. *Transfer Learning*. 2015. URL: <http://cs231n.github.io/transfer-learning/>.
- [8] Andrej Karpathy. *Visualizing What ConvNets Learn*. 2015. URL: <http://cs231n.github.io/understanding-cnn/>.
- [9] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Vol. 1. 2008. URL: <http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html>.
- [10] Modista. *Modista*. URL: modista.com.
- [11] Sameer A Nene and Shree K Nayar. “A Simple Algorithm for Nearest Neighbor Search in High Dimensions”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19.9 (1997), pp. 989–1003.
- [12] Carlton W Niblack et al. “QBIC Project: Querying Images by Content, using Color, Texture, and Shape”. In: *IS&T/SPIE’s Symposium on Electronic Imaging: Science and Technology*. International Society for Optics and Photonics. 1993, pp. 173–187.
- [13] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).
- [14] Neal Khosla Vignesh Venkataraman Amrit Saxena. “Building an Image-Based Shoe Recommendation System”. 2013. URL: <http://cs229.stanford.edu/proj2013/KhoslaVenkataramanSaxena-BuildingAnImageBasedShoeRecommendationSystem.pdf>.
- [15] Neal Khosla Vignesh Venkataraman Amrit Saxena. “Leveraging Computer Vision Principles for Image-Based Shoe Recommendation”. 2014. URL: http://cvgl.stanford.edu/teaching/cs231a_winter1415/prev/projects/CS231AFinalProjectReport.pdf.
- [16] A. Yu and K. Grauman. “Fine-Grained Visual Comparisons with Local Learning”. In: *Computer Vision and Pattern Recognition (CVPR)*. June 2014. URL: <http://vision.cs.utexas.edu/projects/finegrained/utzap50k/>.