UNIVERSITY OF CALIFORNIA, SAN DIEGO

Predicting Facial Attractiveness Within an Individual’s Portraits using Siamese Networks

A Thesis submitted in partial satisfaction of the requirements for the degree Master of Science in Computer Science by

Angel Iek Hou Zhang

Committee in charge:

Professor Gary Cottrell, Chair
Professor Sanjoy Dasgupta
Professor Lawrence Saul

2016
The Thesis of Angel Iek Hou Zhang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

__________________________________________

__________________________________________

__________________________________________

Chair

University of California, San Diego

2016
DEDICATION

To my love, Darwin.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>viii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>ix</td>
</tr>
<tr>
<td>Abstract of the Thesis</td>
<td>xi</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2 Background</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Psychology of Beauty</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Between-person Attractiveness</td>
<td>4</td>
</tr>
<tr>
<td>2.1.2 Within-person Attractiveness</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Deep Neural Networks Overview</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Related Work</td>
<td>9</td>
</tr>
<tr>
<td>Chapter 3 Data</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Collecting the Images</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Preprocessing</td>
<td>18</td>
</tr>
<tr>
<td>3.3 Collecting Labels</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Training, Validation and Test Set</td>
<td>22</td>
</tr>
<tr>
<td>Chapter 4 Model Setup</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Network Architectures and Loss Functions</td>
<td>24</td>
</tr>
<tr>
<td>4.1.1 Convolutional Neural Network (CNN)</td>
<td>26</td>
</tr>
<tr>
<td>4.1.2 AlexNet</td>
<td>27</td>
</tr>
<tr>
<td>4.1.3 Siamese Networks</td>
<td>31</td>
</tr>
<tr>
<td>4.1.4 Loss Functions</td>
<td>32</td>
</tr>
<tr>
<td>4.2 Our Model Architecture</td>
<td>37</td>
</tr>
<tr>
<td>4.3 Training the Networks</td>
<td>41</td>
</tr>
<tr>
<td>4.4 Hyperparameter Optimization</td>
<td>42</td>
</tr>
<tr>
<td>Chapter 5 Results and Discussion</td>
<td>47</td>
</tr>
<tr>
<td>5.1 Network Performance</td>
<td>47</td>
</tr>
<tr>
<td>5.1.1 CNN_A</td>
<td>48</td>
</tr>
<tr>
<td>5.1.2 CNN_BN</td>
<td>48</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1: Warping the faces to make them more or less memorable, as described in Khosla et al. ........................................... 10
Figure 2.2: Faces ranging from less to more attractive across different seriousness levels, as described in Efros et al. ..................... 12
Figure 3.1: Face datasets .......................................................... 16
Figure 3.2: Bounding box formed around the face detected by Face++ (Megvii, Inc., 2013) ......................................................... 19
Figure 3.3: An example of the votes received by each image in a person’s photo collection .......................................................... 21
Figure 4.1: Visualization of the second convolutional layer of ALL-CNN described in Springenberg et al. .......................... 27
Figure 4.2: Visualizing the 6-th (a) and 12-th (b) convolutional layers of the ALL-CNN described in Springenberg et al. ............... 28
Figure 4.3: AlexNet is a convolutional neural network trained to classify natural images from the ImageNet database (Krizhevsky et al., 2012) 29
Figure 4.4: Results of running the MNIST dataset through a siamese network 34
Figure 4.5: Learned mapping of all 10 digits from MNIST after 100 iterations (a) 400 iterations (b) and 50000 iterations (c) ............. 35
Figure 4.6: An example of our siamese network architecture with ranking loss 46
Figure 5.1: Ranking as decided by 9 human workers versus ranking as predicted by our model .................................................. 52
Figure 5.2: Ranking as decided by 9 human workers versus ranking as predicted by our model .................................................. 53
Table 3.1: Matrix of worker’s votes for images used to calculate Krippendorff’s $\alpha$. $i_k$ is the $k$-th image, $w_j$ is the $j$-th worker, and $v_{jk}$ is the vote the $j$-th worker gave to the $i$-th image.
ACKNOWLEDGEMENTS

First and foremost, I would like to acknowledge Professor Garrison W. Cottrell for being such a great advisor. I do not think there is any other advisor out there who can beat his sense of humor, and his easygoing, yet supportive and motivating nature. I am honored to have worked with such a wonderful mentor, and none of this would be possible without his guidance.

I would also like to acknowledge Professor Sanjoy Dasgupta and Professor Lawrence Saul for being on my thesis committee, and for taking time out of their busy schedules to read this work.

I would like to acknowledge my fellow members of the GURU lab. I would like to thank Mark Chen for his help on collecting the dataset, and Linjie Li for her help on preprocessing. I would also like to thank Linjie, Yufei Wang and Sanjeev Jagannatha Rao for the many hours they spent helping me with my work, and for answering my questions, even if some of them came up in the middle of the night. Their help has been so valuable and their words of encouragement have helped me overcome many moments of frustration.

Finally, I would like to acknowledge all the people who have provided me with moral support and love through this time. To my parents: Thank you for sending me to the United States and giving me the opportunity to pursue a better education. I am ever so grateful. To my friends: I am so lucky to have met such great people during my two years here in San Diego. Looking forward to having fun on the weekends with you guys really helped and motivated me to get through a lot of my work. Zachary Meyer, I will miss going to class with you every day, playing Trivia Crack to stay awake and visiting the coffee girls every day. I am grateful to know that there is
someone who will always have my back. Kristjan Jonsson and Ian Chen - you are my two musketeers and it is so great to have buddies who understand that pho, boba and ice cream are the best comfort food after a long week. Wael Farhan and Rana Alotaibi - you two are the most sane and down-to-earth of my friends - thank you for keeping me grounded. I have learned so much about strength and perseverance from you. Callan Yan - you have supported and encouraged me for many years now. Thank you for helping me on this journey, and for really being there no matter how much time and distance has passed. David Weisbart - I learned one of my first proofs from you and you have influenced the way I approach and think about many problems. I will always treasure the summer afternoons we talked about math, life, and nothing at all. Thank you - you are a truly great teacher and an amazing friend. Last but not least, Darwin - thank you for always supporting me, feeding me and keeping me sane throughout these past few years. You made my days brighter even when I was going through the most challenging times, and I could not have done all this without you. Thank you for being my rock and shelter.

Thank you, this project would not be possible without any of you.
ABSTRACT OF THE THESIS

Predicting Facial Attractiveness Within an Individual’s Portraits using Siamese Networks

by

Angel Iek Hou Zhang

Master of Science in Computer Science

University of California, San Diego, 2016

Professor Gary Cottrell, Chair

People often have to decide which photograph of themselves to include in a portfolio or album, or share online. When making this choice, people are usually guided by what they think others will deem attractive. There is some consensus as to which portraits are the most attractive, and which ones are the least attractive. The goal of this thesis is to predict which photo of an individual is the most popular among viewers. In this work, we show experimentally that there is agreement among people regarding the attractiveness of an individuals photo and that it is possible to model and predict which photos are the most popular. We collected a dataset featuring
portfolio pictures of actors and actresses, models and singers. Human votes for these images were also collected. We then trained a siamese network with a ranking loss function to predict the relative ranking of images. Predictions were made on the test set using one side of the network. We compared these predictions with the human votes and were able to achieve an accuracy of 61%.
Chapter 1

Introduction

In this day and age, first impressions are often formed from photographs. Before we ever meet someone in person, we have most likely already seen his/her profile picture on some form of social network such as Facebook, LinkedIn, Google+ or OkCupid. With the advent of self-facing camera phones and a dozen social networking sites on which we can control and market our own brand, we have become more and more obsessed with taking good photographs of ourselves. In 2013, Oxford Dictionaries rated “selfie” to be the word of the year (Oxford Dictionaries, 2013). In 2014, “selfie” was officially added to the Merriam-Webster dictionary (Gilber, 2014) and in 2015, Kim Kardashian sold over 100,000 copies of her book, *Selfish*, which contains 352 pages of her staring sultrily into the camera (Webber, 2015). People spend hours trying to take the perfect photograph and craft the most attractive representation of themselves. This poses an interesting problem - given photographs of the same person, is there one that people can agree on as being more attractive than others, and if so, how can we pick that one?

What makes something beautiful has always been very difficult to pinpoint,
and we often hear that beauty is in the eye of the beholder. However, studies have shown that attractiveness ratings of faces are in fact highly consistent (Cunningham et al., 1995). We mostly agree on what is attractive and what is not, and this makes choosing the right profile picture more meaningful. Automating this process is a useful application for industry and personal use.

Prediction problems fall into the realm of machine learning, and recently, deep neural networks have become a very popular tool that is used in both supervised and unsupervised learning. It has been demonstrated that they work especially well in image-related tasks. The introduction of neural networks revolutionized the field of computer vision and greatly improved image recognition accuracy rates. State-of-the-art networks have an error rate of only 0.23% on classifying handwritten digits from the MNIST dataset, which is even better than human performance (Ciresan et al., 2012). In 2012, a deep convolutional neural network (CNN) was trained on the ImageNet dataset for the image recognition contest ILSVRC-2012, and its top 5 predictions included the true label 84.7% of the time (Krizhevsky et al., 2012). The current state-of-the-art network has 152 layers and its top 5 predictions include the true label 96.43% of the time (classification error of 3.57%) (He et al., 2015). These results really are quite remarkable when we consider that there are 1.2 million images and 1000 categories in this dataset. The networks have been getting deeper every year, as deeper networks can learn more features, and this results in higher accuracy. The advances made in distributed systems and GPU performance have made it possible to train deeper networks on larger datasets, allowing us to reach higher and higher levels of accuracy on image recognition and other machine learning tasks.
Deep networks can have many architectures, each designed for a specific task. One such architecture is the siamese network, which Bromley et al. first introduced for signature verification. A siamese network consists of two identical networks that share weights. Two images are given to the siamese network as input, and in the classical siamese network, each image has a corresponding label that is either 1 or 0. The label 1 indicates that the two images are ‘similar’ and the label 0 indicates that the two images are ‘not similar’. In the original work, the authors used the siamese network to verify whether two signatures are by the same person or not. Their network detected 95.5% of the genuine signatures. The siamese network has since been extended and used for tasks such as face identification. These networks are especially useful in part because once they have learned a similarity function, they can be used with an unbounded number of unlabeled images. The goal of this project is to use a modified siamese network to determine which photograph is more attractive.

Chapter 2 reviews some related work and gives an overview on the theoretical background needed in this project. Chapter 3 describes the data collection process and Chapter 4 describes the model setup in detail. Chapter 5 gives a discussion and analysis of the results. Chapter 6 provides a summary and discusses possible topics to be explored in the future.
Chapter 2

Background

2.1 Psychology of Beauty

2.1.1 Between-person Attractiveness

Humans have been intrigued by beauty since the beginning of time. To this day, people still remember Helen of Troy, a woman who lived over 3000 years ago, as the most beautiful woman in Greece (Encyclopedia Brittanica, 2015). The Ancient Greeks were the first known people to develop a theory of aesthetics, a topic in philosophy that is still being studied today (Konstan, 2015). The Pythagoreans from the 6th century BCE believed that order and symmetry make things beautiful (Leddy, 2012), and relatively recent studies corroborate this point of view (Grammer et al., 1994). Other factors have also been shown to influence people’s opinions on whether a person is attractive or not. Several studies have found that even from looking at just one photograph, observers judge women who are wearing makeup more positively than those who are not. Women with cosmetics are associated with more positive
traits and more prestigious professions (Richetin et al., 2007). Not only do people see attractive individuals in a more positive light, they also judge these individuals’ personalities more accurately (Lorenzo et al., 2010). This accuracy measure is a comparison of the observers’ ratings with the individuals’ own self-ratings, and this has the implication that more attractive people can better manipulate what other people think of them. Although it is often said that beauty is in the eye of the beholder, there is actually high cross-cultural agreement on which faces are the most attractive (Cunningham et al., 1995). Since people cannot really change the faces they are born with, it is more interesting and realistic to investigate how they can modify the way they are presented and viewed without undergoing drastic surgical procedures.

2.1.2 Within-person Attractiveness

There are ways for a person to manipulate his/her appearance in order to be more appealing. Gueguen et al. showed that when a waitress wore makeup, she was tipped significantly more than when she did not. It was also shown that in a bar setting, when two women wore cosmetics, they were approached more often by men than when they did not (Guguen, 2008). This effect extends to still photographs as well. Although female opinion was not altered, males rated photographs of women to be more attractive when they were wearing makeup (Cash et al., 1989). Axelsson et al., 2010 showed that photographs of people who had been sleep-deprived were rated to be less attractive than their photographs that were taken after a normal night’s sleep. It has been found that even ovulation, a phenomenon of which we are usually not consciously aware, can be picked up from a photograph (Bobst et al.,
This particular study averaged 25 photographs that were taken during the ovulation phase and 25 photographs that were taken during the luteal phase of the same women to create a fertile and a non-fertile prototype. These prototypes were then used to transform 20 photographs of new women. Male observers were presented with pairs of images - one that was transformed using the fertile prototype and one that was transformed using the non-fertile prototype. They were then asked to rank which face is more attractive, more caring, more flirtatious, and more likely to be open to a date. In every category, the faces that were transformed with the fertile prototype were chosen significantly more frequently. Most of the studies done so far involve the attractiveness ratings of females by males, possibly because these results are more pronounced than others. As discussed above, female observers’ ratings were unaffected by whether the women in the photographs were wearing makeup or not. The difference is most likely due to evolutionary factors that cause men to place more value on physical appearance, although there could also be other psychological effects involved. This does not mean that rankings of male attractiveness are insignificant; women’s preferences are perhaps more subtle. It has been shown that women find symmetrical and masculine male faces to be more attractive, but their preferences vary with their menstrual cycle (Gangestad et al., 2005).

Being able to represent ourselves with attractive photographs is not merely an act of vanity, as it truly impacts other people’s opinions of us. If attractive people get evaluations that are more consistent with their self-evaluations, then it should be true that if we can present a more attractive self, we will have more control over what other people think of us.

Since attractiveness is such a huge factor in what other people think of us,
and we are often judged by our profile photos in this digital age, it makes sense to study this topic and learn how to portray ourselves in the most attractive way possible. In the next section, we will discuss deep neural networks, the foundation of our model. We will explore some of the history of deep neural networks, their impact, as well as their relevance to our goal of choosing the most attractive photograph of an individual.

2.2 Deep Neural Networks Overview

Deep networks are embedded in many aspects of our everyday lives. Many intelligent mobile personal assistants such as Siri and Cortana rely on long short-term memory recurrent neural networks (LSTMs) for speech recognition (Graves et al., 2013). Many companies are competing for the most cutting-edge technologies for speech recognition, natural language processing and image classification and most of these are powered by deep learning. With big companies like Apple acquiring Perceptio and VocalIQ, Facebook acquiring Wit.ai, and Google acquiring DeepMind - all deep learning startups - for millions of dollars, we can see that deep learning is without a doubt relevant in today’s industry. It is also easy to see why our problem, which involves analyzing people’s profile pictures, is relevant to companies such as these. In fact, face-related tasks are already part of these companies’ latest developments. Emotient, a deep learning company that analyzes people’s facial expressions with the goal of targeted advertising, was also recently acquired by Apple. With potential benefits such generating more advertising revenue, it is easy to see why facial expression analysis is of interest to companies. It is also natural for companies, as well as researchers, to care about facial attractiveness. From the industry standpoint,
social and professional networks, as well as dating services like eHarmony, OkCupid, and Match.com can easily benefit from facial attractiveness analysis. Researchers are also interested in the cognitive and psychological aspects of what we find attractive in a face, and a predictive deep learning model might offer some insight.

So when exactly did deep networks become so popular in industry and research? The first deep networks actually date back to the 1960s, when a deep feed-forward multi-layer perceptron was first introduced (Ivakhnenko, 1968). A feed-forward net is essentially a chain of matrix multiplications composed with linear or non-linear functions. The convolutional network architecture dates back to 1980 and max-pooling dates back to the early 90s (Fukushima, 1980; Weng et al., 1993). Back-propagation, a gradient-descent method for updating the weights in a neural network with respect to a specified loss function, was first applied to deep networks in the late 80s (LeCun et al., 1990). Even though all these tools existed over half a century ago, deep learning really did not gain wide-spread popularity until the 2000s. Their unpopularity can be largely attributed to the lack of computational power at the time. With the onset of fast and cheap GPUs, as well as vastly improved distributed systems and parallel programming, deep learning was revived, first in image classification, and then in any field that was able to frame their problem as a learning problem. Deep learning really took off after a deep network won the image classification contest in 2012 (Krizhevsky et al., 2012). This achievement showed everyone the potential power of deep learning, and the race to invent even better deep networks and applications has become much more competitive ever since.

The deep learning framework is the best tool we currently have for a learning problem involving images. The great successes of deep learning models in the image
classification domain have inspired us to apply it to our own face images problem.

2.3 Related Work

Recent works have revealed many interesting things about our perception of faces. Khosla et al. showed that slight alterations to a photo of someone’s face can make him/her more memorable. Memorability was measured using experimental results from human performance on specific memory tasks. The authors first represented the original faces in a low-dimensional feature space using facial landmark-based annotations. They then defined a cost function that penalized faces that deviated too much from the original identity, age, attractiveness and other facial attributes, and faces that did not reach their memorability requirement. They then generated new faces by warping the original faces. This warping involved randomly sampling points that were close to the original projection, and from the random sample, they chose the face that minimized the cost function. Following this procedure, they were able to construct faces from the original photographs that had higher or lower memorability with minimal adjustments to identity and other facial attributes. An example of the original and warped faces can be seen in Figure 2.1. A recent work developed a method that is able to modify an individual’s age and facial expressions (Gardner et al., 2016). The researchers’ method involves deep manifold traversal. We will omit the specifics as they are beyond the scope of this project, but their method follows a framework similar to Gardner et al. - the face is projected onto a lower dimension, the result is warped and the face is reconstructed by optimizing with respect to a loss function. The ability to warp face images to change a viewer’s perception has some interesting implications - one can imagine that this technique
Figure 2.1: Warping the faces to make them more or less memorable, as described in Khosla et al.

can be extended to create more attractive portraits. In the future, this could allow people to perform digital surgery’ automatically, without using traditional Photoshop or other photo editing techniques that take time and skill.

Research focused on developing models of facial attractiveness has also been progressing. We will visit three studies that are closely related to ours. In a 2006 study, some classical machine learning models were used to predict facial attractiveness. Eisenthal et al. collected a two datasets of young Caucasian women of varying attractiveness levels. The datasets were created in a highly controlled environment, with the same lighting conditions. Each image showed a face with a neutral expression. Ratings were collected from 28 and 18 raters respectively, and each image was
ranked on a scale of 1 to 7, where 7 is very attractive and 1 is very unattractive. Feature landmarks such as the face length or eyebrow thickness were recorded and the faces, as well as these feature landmarks, were then projected into a lower dimensional space using principal component analysis. The authors then used K-nearest neighbors (KNN) and support vector machines (SVM). Using KNN, they were able to achieve a highest accuracy of 86% on the handcrafted feature landmarks, and using SVM, they were able to achieve a slightly lower accuracy of 84%. This was one of the first studies to show that machines can learn to predict facial attractiveness.

Gray et al. created a model that learns and predicts beauty in female faces without using facial landmarks. They collected images from the website Hot or Not, and recruited 30 workers to view pairs of images and pick the more attractive one. They then formulated an exponential cost function and used gradient descent to minimize this cost. The authors designed and tested 4 models. The first one was an eigenface model, which is a more classical method that uses singular value decomposition and linear filters to extract features and predict which images are more attractive. The second, third and fourth models are more standard machine learning models, using neural networks with various configurations. The evaluation metric for their models used Pearson’s correlation coefficient to measure the correlation between the predicted rankings and the human rankings. Using their best model, they were able to achieve a 0.458 correlation. This study showed that it is possible to create a model that does not use landmark features and predicts attractiveness ratings that significantly correlate with human ratings.

A study of within-person attractiveness, perhaps the most similar to our project, was published in 2014 (Efros et al., 2014). Efros et al. were also interested
in portrait attractiveness, and in particular, which expressions make a face most attractive. They collected their dataset by capturing videos of their subjects watching a collection of short videos. They were able to collect 12 minutes of footage from each of their 11 subjects. Still-frames were cropped from these videos using a greedy algorithm to pick 200 to 250 images with unique expressions. They then extracted histogram of oriented gradients (HOG) features, which are thought to capture the properties of facial expressions well. The researchers also asked workers to view pairs of images depicting the same person and choose the more attractive one, or the more serious one. A score was assigned to each image using the Bradley-Terry model and trained regression models to predict the scores for each image. Examples of the portraits of different attractiveness levels can be seen in Figure 2.2. They used support vector regression and gradient boosted regression trees to predict the attractiveness and seriousness scores of each subject, and they found that they were able to achieve an error of around 0.06 with either method. The error metric they used calculates the deviation of their predicted attractiveness ranking from the actual attractiveness ranking, averaged over a range of seriousness scores. The researchers also explored

![Figure 2.2](image_url)  

**Figure 2.2:** Faces ranging from less to more attractive across different seriousness levels, as described in Efros et al.
between-person rankings and were able to achieve similar results.

Our project differs from Efros et al. in several ways. First, our dataset is not controlled. Our goal is to choose a most attractive selfie, which are all taken in the real world, so using images that have been taken in the exact same conditions would not provide us with a representative dataset. Second, Efros et al. used still-frames from videos as their images, which means that any captured moment could be very unattractive. Our dataset, on the other hand, is designed to consist of images that a person already considers to be an attractive depiction of himself/herself. Third, Efros et al. focused on the exact expression that is attractive, so they only collected data from 11 people. Since they were specifically looking at facial expressions, they extracted out only the features that they needed, such as the areas around the eyes and mouth. Although facial expressions will definitely play a part in whether an image in our dataset is considered attractive or not, we will require more data than that if we wish to learn the more general traits of an attractive selfie/portrait. Fourth, rather than apply the classical machine learning models, we will be designing a deep learning model and evaluating its performance on facial attractiveness predictions.
Chapter 3

Data

Due to the data-driven nature of deep learning models, it was necessary for us to invest some time to find and create the right dataset. After training the model on the training dataset, we would want the model to be applicable on samples it has never seen before. The quality and composition of the training data has a large influence on the quality of the learned model. We can imagine some extreme scenarios where over 70% of the original MNIST database are labeled incorrectly, or where we have only 2 examples of each digit - in these situations, it would be next to impossible for any model to learn useful features for generalization. What is interesting, however, is that in order to reflect the statistics of the real world, a dataset must not be too clean. As LeCun et al. noted, the noise in their handwritten digit database is an important feature, as all real-world databases will contain bad examples. In this chapter, we will describe the three stages of our dataset creation process: (1) collecting a dataset of people’s photographs, (2) preprocessing the images according to our needs and (3) gathering appropriate labels for these images.
3.1 Collecting the Images

There are several preexisting face datasets, but unfortunately, none of them suited the purposes of this project. A well-known dataset is the 10k US Adult Faces, shown in Figure 3.1a. There are many datasets such as this one, which contain many images in total, but only have 1 or 2 images of the same person. Without having access to multiple pictures of each person makes it impossible to study within-person attractiveness. Other datasets like CMU Multipie, shown in Figure 3.1b, do have multiple images of each person. However, these photographs were taken in a very controlled setting, and there is not much diversity in the people depicted. Since the goal is to be able to choose the best selfie, such images are not very useful.

Since we were unable to find a satisfactory dataset, we decided to scrape the web for our own. One thing we have to consider when training a model is the size of the dataset. We needed to collect enough data so as to be able to generalize to new, unseen data. Researchers have found theoretical bounds for how much data is required for certain network specifications. Baum et al. derived that a feedforward network with \( N \) linear threshold units and \( W \) weights requires

\[
m \geq O\left(\frac{W}{\epsilon} \log \frac{N}{\epsilon}\right)
\]

examples, where \( 0 \leq \epsilon \leq \frac{1}{8} \). Furthermore, \( 1 - \frac{\epsilon}{2} \) of the \( m \) examples must be predicted correctly during training if we are to arrive at a network that has a probability of \( 1 - \epsilon \) for predicting the correct label on a new example. However, these theoretical bounds do not have much practical use. In practice, the number of examples required for learning a model within an acceptable range of error rates varies depending on the problem domain, and in reality, the amount of data researchers use depends on their intuition gained over time, and by the financial and computational costs that they can afford.
Since the original objective of this project was to create a model that helps people choose their most attractive selfie, the ideal dataset would be created from actual selfies that people post on social networks. In particular, as we are interested in within-person attractiveness, we wanted multiple photographs of the same person. We were not necessarily looking for images in which the people faced directly forward, with their heads straight, since this is not reflective of most selfies. In fact, reviews have found that people more often tend to present their left cheek in portraits (McManus, 1973). This was found to be consistent in selfies, but interestingly enough, people have a right cheek bias when they take selfies of their mirror reflec-
tions (Bruno et al., 2015). Taking this into consideration, we decided that we only needed the photos to have clearly visible faces.

Selfies that already exist on the internet would form the most realistic representation of future, incoming data. However, there are many complications involved in scraping a dataset from social networking sites. The photographs are not always publicly viewable, and even when they are, we cannot guarantee that we can collect enough pictures of the same person. We ended up collecting a database of 204,317 images from the website StarNow, which features the portfolios of aspiring actors, actresses, models and singers. Although this may not be the most natural database to use for our particular task, we note the following in the hopes of also addressing possible concerns regarding this dataset:

1. The images are of actors, actresses, models and singers, and this might influence people’s opinions of them. It is true that these images do not represent the average person’s selfies, but the people depicted in these portfolios are aspiring stars, and not known celebrities, so we have not introduced a potential effect of bias due to celebrity status.

2. Most of the people in the database are more attractive than the average person. As discussed above, this should not make a difference to our task, as the comparison is within-person and it should not be based on their attractiveness relative to other people.

3. Most of the images are high-quality photographs taken by professionals. Although this was initially seen as a disadvantage as it is not representative of real selfies, it might actually be a desirable feature. If all the images are high-

quality and have good lighting, then the attractiveness of the face really becomes the only focus.

We used Selenium, a Python package to scrape the website of interest. The link to each person’s profile was followed and all the photographs on that profile were retrieved. We were able to scrape a total of 204,317 photographs from 53,461 different people’s profiles.

3.2 Preprocessing

In order to make the dataset usable for our problem, we had to first filter out the images that contained no faces or more than one face. We used the Face++ Research Toolkit, a cloud-hosted service that can be used for face detection and landmarking, among other tasks (Megvii, Inc., 2013). After performing this step, we found that face detection failed on most of these images. It could be that Face++ failed to sense some faces, or that many images indeed did not depict any faces. We also filtered out images that were under 2MB in size, so as to ensure that we had high resolution images. After filtering, we were left with 7,929 images of 480 people. Figure 3.2 shows an example of the bounding box that Face++ creates upon detecting a face. From these face images, we filtered out all the people who had fewer than 9 photographs. Our motivation for doing this is that we need a sufficient number of images per person so that finding a better or worse picture is actually meaningful. Of course, 9 may not be the best number, but it should serve as a good starting point. From the 424 people that were left after filtering in this manner, we randomly picked 9 of their images to include in the final dataset of 3,816 images. We processed the
final images by centering and padding them with white on each side, so that each resulting image is square. We then scaled each image to be $256 \times 256$.

### 3.3 Collecting Labels

We collected the labels for our dataset using Amazon Mechanical Turk workers. In order to guarantee the quality of the collected data, we used only workers who had an approval rate of 90% and higher on all their previously completed tasks. The ranking of a single person’s photograph collection constituted a single trial. During each trial, a worker was given the following instruction: There are 9 images of the same person below. Select the three images that you think are most attractive. On the same screen, they were shown 9 images, in a $3 \times 3$ grid. The experiment was designed
so that exactly 3 images needed to be chosen in order for the worker to proceed to the next trial. We decided to have workers pick the top 3 images rather than rank all 9, since this is probably an easier task, with slightly less variation between workers. Each trial took a worker 2 minutes and 4 seconds on average to complete and each person was ranked by 9 distinct workers. There were 80 unique workers in total, however, 89% of the dataset was labeled by 40 of these workers. The low number of workers means that our dataset might be biased toward their particular taste, but again, we thought that this could be a good starting point. Once we had collected the labels, we tallied up the number of votes per image, and used this value from 0 to 9 as our final ranking of a particular image. We thought that this is a reasonable way to rate the images, since the score represents how popular a particular image is. An example of the number of votes a person’s photos received is shown in Figure 3.3.

**Reliability of Data**

Upon visual examination of the data, we noticed that there seem to be some agreement between voters. We were able to find clear clusters showing high number of votes for attractive photos and essentially zero votes for the unattractive ones.

In order to measure if there is actual consistency between the human rankings, we used Krippendorff’s alpha, a statistic often used to measure the degree of agreement between voters and the reliability of collected data (Hayes et al., 2007). In order to compute Krippendorff’s alpha, a reliability matrix is first formed, where each column represents a unit that is being assigned a value (an image in our case, $i_k$), and each row represents a coder’s votes (the worker, $w_j$). The $jk$-th entry is the ranking the $j$-th worker gave to the image $k$. The matrix is shown in Table 3.1.
Figure 3.3: An example of the votes received by each image in a person’s photo collection

Krippendorff’s alpha takes the following form:

$$\alpha = 1 - \frac{D_0}{D_E}$$

The calculation of $D_O$ and $D_E$ depend on the values in the reliability matrix and the form of the data. The details will be omitted, but $D_O$ represents the observed

Table 3.1: Matrix of worker’s votes for images used to calculate Krippendorff’s $\alpha$. $i_k$ is the $k$-th image, $w_j$ is the $j$-th worker, and $v_{jk}$ is the vote the $j$-th worker gave to the $i$-th image.

<table>
<thead>
<tr>
<th></th>
<th>$i_1$</th>
<th>$i_2$</th>
<th>...</th>
<th>$i_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_2$</td>
<td>$v_{21}$</td>
<td>$v_{22}$</td>
<td>...</td>
<td>$v_{2n}$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$w_m$</td>
<td>$v_{m1}$</td>
<td>$v_{m2}$</td>
<td>...</td>
<td>$v_{mn}$</td>
</tr>
</tbody>
</table>
disagreement and $D_E$ represents the expected disagreement between voters.

We calculated Krippendorff’s alpha by forming a $80 \times 3,816$ matrix, where each row represented a worker and each column represented one image. Most of the entries constituted missing data, since only 9 workers actually rated each image. Krippendorff’s alpha was chosen in part because it is able to handle these cases where there is a lot of missing data. For the other non-empty entries in the matrix, the value is binary - 1 if the voter selected the image as one of the top three, and 0 otherwise. We found $\alpha = 0.21$, which makes our data unreliable by Krippendorff’s standards (Swert, 2012). However, Swert also notes that even if there is very little disagreement between the voters, Krippendorff’s alpha is low for variables with binary values where one value is rare. This is probably the case with our dataset, since each worker could only pick three images, and most of the entries in the matrix are zeros. It is also likely that we would need to have more workers ranking each image collection if we would like to see clearer results.

3.4 Training, Validation and Test Set

The input into siamese networks is a pair of images, with their respective labels. More details about siamese networks will be given in the following chapter, but this architecture is what motivated us to create a dataset of pairs of images. For each of the 424 people, we formed 36 pairs from their 9 photographs. These pairs of images were labeled with their corresponding scores, as described in the section above. We formed a total of 15,264 pairs. As with other machine learning algorithms, we needed to split our dataset into a training, validation and test set. The training set consists of examples from which the model will learn, the validation set is for testing
different models and fine-tuning the model hyper-parameters, and the test set is for
the final test once a model has been picked, to see how well the model generalizes.
We made sure that the identities in the training set, validation set and test set were
different. In other words, an image in the test set depicts a person who has never been
seen before in the training or validation set. We created a split so that the test set
made up 30\% and the training and validation set made up 70\% of the original dataset.
We split the 70\% further into approximately 85\% training and 15\% validation data.
These splits resulted in 9,000 pairs of 250 people for the training set, 1,692 of 47
people in the validation, and 4,572 of 127 people in the test set.
Chapter 4

Model Setup

In this chapter, we will give some background on neural networks and describe our model. We will discuss our chosen network architecture as well as the details involved in training, optimizing and fine-tuning the network.

4.1 Network Architectures and Loss Functions

Currently, there are no hard-set rules for how to set up an optimal network architecture. The design of high-performance neural networks has been mostly empirical. Similar to the process of using other machine learning algorithms, the error rate on the validation set is often used as an indication of which model is performing better, and with what parameters. There are many decisions to make when designing a network - what kind of layers to include in what order, how many layers to have, and how many units to include in each layer are just a few such decisions. There are various points of view as well – the researchers who designed the winning 152-layer network for the ILSVRC contest last year believe that deeper networks gives better
results (He et al., 2015). However, (Ba et al., 2014) investigated whether networks have to be deep in order to achieve high accuracy rates, and for a few specific tasks they tried, they were able to get similar results using shallow nets. Even if we have chosen a good architecture, there is still the problem of learning the best weights. (Albertini et al., 1993) showed for some neural networks that the weights for a global minimizer are unique. So given a network, there is only one global minimizer, but the gradient-descent type methods used to train the network make no guarantees about achieving a global minimum. In fact, the local minimum that the methods discover depends heavily on initial conditions and learning rates. The existence of a unique global minimizer makes it unlikely that our current training methods will achieve the best possible result. In practice, however, we are able to get good enough results converging to local minima. Since coming up with a good network architecture can be difficult and time-consuming, we decided to take inspiration from AlexNet, the winner of the 2012 ILSVRC competition. For our task, we do not actually have absolute ratings of the images. Each image was assigned a score, but this score is a relative ranking. We decided to predict how the images rank relative to each other with a siamese network, where each side is a convolutional neural network, using the ranking loss function introduced in (Wang et al., in review).

In the sections below, we will give some background and details on convolutional neural networks and siamese networks and review some commonly used architectures.
4.1.1 Convolutional Neural Network (CNN)

In recent years, convolutional neural networks (CNNs) have become very popular. CNNs were designed specifically for image inputs. Before the arrival of CNNs, neural networks had a difficult time dealing with large images, since every neuron in a fully connected layer was connected to every pixel in the input image, and this made for an enormous weight matrix. CNNs relieve each neurons of having so many connections. CNNs are bio-inspired networks – human vision is actually a picture consisting of many small patches that are pieced together by our brain. Each of these small patches is called a receptive field – this is the region in our visual field for which a particular group of visual cortex cells are responsible (Thomas, 1970). CNNs work in a similar manner. An intuitive way to understand convolution is that it has a smoothing or blurring effect on the object on which it acts. A convolutional layer has a number of kernels with

- size: $k \times k$
- stride: $s$
- padding: $p$

Each kernel is analogous to a group of visual cortex cells, and each element in the $k \times k$ matrix represents a weight. Suppose we have an $n \times n$ image, with 3 channels. The image data on each color channel is convolved with the kernel. The $k \times k$ kernel is moved over the $n \times n$ image channel with a stride of $s$. At each step, it is multiplied element-wise with $k \times k$ local image pixels. Finally, the resulting values are summed across all three channels to get a single channel.
CNNs are inspired by the human vision system, and upon examination of their learned features, it is quite exciting to note that they fit with our previous hypotheses about image representations in our brain. Figure 4.1 shows that the features learned in the first 2 layers are edges and colour gradients. Features learned in the higher layers are more complex. Using deconvolution, we are able to visualize the features learned in the higher layers of a CNN (Matthew D Zeiler et al., 2014). As seen in Figure 4.2, the higher convolutional layers learn features like intricate patterns, eyes and even representations for dogs’ faces. The high performance of CNNs on vision-related tasks makes their structure suitable for our problem. Below, we will give some details on AlexNet, a widely used CNN.

4.1.2 AlexNet

AlexNet was the winning entry in the 2012 ILSVRC contest. Although CNNs had been around for a while before that, they were not really known in the field
Figure 4.2: Visualizing the 6-th (a) and 12-th (b) convolutional layers of the ALL-CNN described in Springenberg et al.

of computer vision. In the 2012 contest, AlexNet beat the runner-up entry with a 10% better top-5 error rate. It was trained on a subset of ImageNet that has 1,000 classes and approximately 1,000 images per class. It achieved a 15.3 top-5 error rate, meaning that in its predictions on the test data, the correct class label was within the top 5 most likely labels given by the model 84.7% of the time.

Convolutional layers had existed for a while, but AlexNet was one of the first neural networks to really take advantage and use multiple convolutional layers. The 2013 winning entry, ZF Net, was just a modification of AlexNet. The number of units in the layers, the stride and kernel size, as well as the hyper-parameters were tuned and resulted in a 1.7% improvement. This example shows us that there is indeed value in modifying and fine-tuning an existing network for our own task. Although
networks with better performance have been designed since then, those networks are generally much deeper and require more computational power to train. Since AlexNet is relatively easy to modify and train, while giving good results, we have decided to use this as inspiration for our base architecture.

![Figure 4.3](image)

**Figure 4.3**: AlexNet is a convolutional neural network trained to classify natural images from the ImageNet database (Krizhevsky et al., 2012)

AlexNet comprises of 5 convolutional layers followed by 3 fully-connected layers and a softmax layer with 1000 outputs. The schematic of the architecture is shown in Figure 4.3. As we discussed above, convolutional layers use kernels that convolve with the input of each layer, forming sparse connections between each layer’s inputs and outputs. Fully-connected layers form dense connections, with weight connections between the input and every single neuron in the layer. Softmax layers are commonly used as an output layer. The softmax function,

\[
\sigma(z)_i = \frac{e^{z_i}}{\sum_k e^{z_k}}
\]

gives an output that can be interpreted as the probability that label \(y_i\) is the correct label for the input. The outputs in the 1000-way softmax layer sum up to 1, so the top-5 error is found by comparing the classes of the 5 softmax layer outputs with
the highest values to the actual class. AlexNet also uses normalization and pooling. The normalization layers in AlexNet are local-response normalization layers, which are biologically inspired and cause neurons from different kernels to compete. The addition of these layers improved their accuracy by 1%. The pooling layers in AlexNet have a stride of 3 and a kernel size of 2, and perform max-pooling. What max-pooling does in this scenario is decrease the size of the output by taking the maximum value in a $3 \times 3$ region, then moving the kernel 2 steps over to take the maximum value in the next $3 \times 3$ region. This method of overlapping pooling helped further improve their results by approximately 0.3%.

The input into AlexNet is a $224 \times 224$ image with 3 color channels. The first convolutional layer has 96 kernels, each with size $11 \times 11 \times 3$ and stride 4, and performs convolution on the input image. The output of this layer is passed through the local-response normalization and max-pooling layers, and is then given as input to the next convolutional layer. The second convolutional layer has 256 kernels, each with size $5 \times 5 \times 48$, and performs convolution on its input. Then, normalization and pooling is performed again. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ and the fourth convolutional layer has 384 kernels, each with size $3 \times 3 \times 192$. The fifth convolutional layer has 256 kernels, each with size $3 \times 3 \times 192$, and is followed by a pooling layer. Following the convolutional layers are 2 fully-connected layers, each with 4096 neurons, and the 1000-way softmax layer. The non-linear activations used in the convolutional and fully-connected layers are Rectified Linear Units (ReLU).
4.1.3 Siamese Networks

Since our scores are artificial rankings, and in reality, attractiveness rankings do not work on such a discrete spectrum, we found that the siamese network architecture with a modified ranking loss is more suited for our needs.

Traditionally, siamese networks have the advantage of not requiring the number of classes to be known in advance. They are also good at dealing with applications that have an enormous amount of classes with only a few examples each. One such application is face recognition, where we have lots of identities, but only a few images of each person. The problem we are trying to solve is somewhat similar to this. There are many identities, and we only have a few images per person, and within each person, we would like to find a ranking of their images. The classical siamese network by itself cannot give a ranking, but its architecture matches the conditions of our problem. Below, we will describe the structure of classical siamese networks.

The siamese network is named after siamese twins, who are physically joined at birth. This network is an architecture that contains two identical copies of the same network, with shared weights. It takes pairs of images as input, with corresponding labels of 1 or 0, identifying whether a pair is similar or not similar. This is what makes the model so simple - nothing other than similarity information is needed. Each image is then fed through one side of the network (recall that both sides are identical and share weights). The loss is then calculated and backpropagation occurs as it does in any regular neural network (Bromley et al., 1993)
4.1.4 Loss Functions

The task that the siamese network solves is therefore very closely linked to its loss function. The general goal of the siamese network is to be able to look at a new data point and tell if it matches with something it has already seen, or if it is different. This testing is typically done using one side of the siamese network, and the results can be attained in various ways. In one of the first applications of the siamese network, Bromley et al. verified people’s signatures by constructing a model of their signatures. During testing, a representation of a new signature example was created by using the output of one side of the siamese network as the final feature vector. This output vector was then compared to the model’s multivariate normal density representation for that person’s signature to determine whether the new signature was genuine or a forgery. Notice that this model still depends on knowing the identity of the person whose signature we are trying to verify, so that the new signature could be compared to the right person’s signature model (hence verification, and not identification, which is a much larger problem). In 2005, the siamese network was extended to face verification by using a different loss function (Chopra et al., 2005). In this paper, the authors introduced an energy-based loss function, which assigned low energy to similar pairs, but high energy to dissimilar pairs. The loss function takes the following form:

\[ L(W) = \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i) \]

where \(X_1\) and \(X_2\) is a pair of images, \(Y\) is a label that takes value 0 if the pair is the same and 1 if the pair is different, \(W\) is the shared weights and \(P\) is the number of
all pairs. The loss of a single example is as follows:

\[
L(W, (Y, X_1, X_2)^i) = (1 - Y)L_S(D_W(X_1, X_2)) + YL_D(D_W(X_1, X_2))
\]

\[
= (1 - Y)\frac{2}{Q}(D_W)^2 + (Y)2Qe^{-\frac{2D_W}{Q}}
\]

where \(L_D\) and \(L_S\) are the partial loss functions for a dissimilar and similar pair respectively, \(D_W = \|G_W(X_1) - G_W(X_2)\|\) with \(G\), any function that is differentiable with respect to \(W\), and \(Q\), the maximum possible value of \(D_W\). Notice that the first term of \(L(W, (Y, X_1, X_2)^i)\) is strictly increasing, and the second term is strictly decreasing, so this loss function captures the correct behavior of increasing the energy of dissimilar pairs and decreasing the energy of similar pairs. The main difference between this model with the energy-based loss function and the signature verification model is that this one does not attempt to estimate the probability distribution for each input category, so the architecture of each side of the siamese network is a lot more flexible, since normalization is not necessary. However, this model can still only be used for verification. The same authors modified this loss function further in order to achieve the goal of dimensionality reduction, so that similar pairs would be mapped closely and dissimilar pairs would be mapped further away from each other in the output space, without having any distance information about the pairs in the input space (Hadsell et al., 2006). This learned mapping can generalize to new, unseen examples. The modified loss function takes the same form as above, except \(D_W(X_1, X_2) = \|G_W(X_1) - G_W(X_2)\|_2\), \(L_S(D_W) = \frac{1}{2}(D_W)^2\) and \(L_D(D_W) = \frac{1}{2}(max(0, m - D_W))^2\), where \(m\) is some positive margin. This loss function also gives the desired behavior, and as termed by the authors, the \(L_S\) term causes similar pairs
to attract and the \( L_D \) term causes dissimilar pairs to repel. Just by changing the loss function, the siamese architecture now gives us another type of solution. One of the experiments they did was attempt to have the model create a 2-dimensional mapping of the subset of the digits 4 and 9 from MNIST. Figure 4.4 shows this projection in 2D, and we can see that although the model was given no prior information other than whether the digits were the same or different, it was able to learn a mapping that separates the 4s from the 9s, and clusters them. Figure 4.5 shows the learned mapping of all 10 digits from the MNIST dataset at various stages in training.

**Figure 4.4:** Results of running the MNIST dataset through a siamese network

**Ranking Loss Function**

We’ve seen how important the loss function is for siamese networks, but none of the ones described above are suitable for our problem. Instead, we had to use the
Figure 4.5: Learned mapping of all 10 digits from MNIST after 100 iterations (a) 400 iterations (b) and 50000 iterations (c)
following ranking loss, introduced in Wang et al.:

\[
L = \begin{cases} 
\frac{1}{2} \max(0, |D_W| - m_s)^2 & \text{if } D_g < m_s \\
\left\{ \frac{1}{2} \max(0, m_s - D_W)^2 + \max(0, D_W - m_d)^2 \right\} & \text{if } m_s \leq D_g \leq m_d \\
\frac{1}{2} \max(0, m_d - D_W)^2 & \text{if } D_g > m_d 
\end{cases}
\]

where \( D_g = Y_1 - Y_2 \) is the difference between the true scores of the input pair, \( D_W = G_W(X_1) - G_W(X_2) \) is the difference in the predicted scores for the input pair, and \( m_s \) and \( m_d \) are the margins for similar and dissimilar pairs respectively. Notice that when \( D_g > m_d \), the loss has the same form as the partial loss function for dissimilar pairs shown in the previous section (Hadsell et al., 2006). The difference is that \( D_W \) in Wang et al. is the difference between the predicted scores of the image pair, whereas \( D_W \) in Hadsell et al. is the Euclidean distance of the outputs. This distinction is important, and is what makes it possible for us to use the siamese architecture to solve our ranking problem. Consider only the case when \( D_g > m_d \), that is to say, the true score of image 1 is greater than the true score of image 2. There are two possible situations: if the difference between the predicted scores is greater than the margin, then there should be no loss incurred, otherwise, the loss should be positive. In other words, when \( D_W > m_d \), we know that means that the predicted score of image 1 is greater than the predicted score of image 2 by at least our desired margin \( m_d \), and since the predicted higher score corresponds to the correct image with the true higher score, the model has done well and there should be no penalty. On the other hand, when \( D_W \leq m_d \), we know that means that the predicted score of image 1 is either less than the predicted score of image 2, or is not greater than the predicted score of image 2 by enough of a margin, so the higher prediction does not
correspond to the right image with the higher true score, or does not associate with it strongly enough. The other cases can be similarly understood. The margins $m_d$ and $m_s$ are what allow us to compensate for the amount of variance in the collected data.

### 4.2 Our Model Architecture

We follow Chopra et al. in using two identical CNNs in our siamese network. As the authors pointed out, CNNs have the advantage of learning optimal shift-invariant local feature detectors and build representations that are robust to geometric distortions of the input image. This is well-suited to our problem. For each CNN, we used AlexNet as a starting point. We removed the local-response normalization layers and the fourth and fifth convolutional layers, and we modified the fully-connected layers. We will describe the architecture of our model in detail below.

**Image Layer**

The input to the CNN is a $227 \times 227$ image with 3 color channels. We augmented our data by mirroring, and we normalized the data using the ImageNet mean file. Mirroring helps increase our training dataset and normalization is useful as it can get rid of the effects of extremely dark or extremely bright images that might dominate otherwise.

**Layer 1: Convolutional Layer**

The first convolutional layer has the same structure as that of AlexNet, i.e. with 96 kernels, each with size $11 \times 11 \times 3$ and stride 4. We also used the same Rectified
Linear Units (ReLU) as our activation functions. Since the ReLU function takes the maximum of 0 and \( x \) given some input \( x \), it zeros out many of the output values, resulting in sparse activations. The output feature map has dimension \( 55 \times 55 \times 96 \).

**Layer 2: Pooling Layer**

This layer performs max pooling with a kernel size of \( 3 \times 3 \times 96 \) and a stride of 2. This pooling overlap allows us to retain some more information than ordinary pooling with no overlap, and results in slightly higher accuracy. The output feature map dimension is \( 27 \times 27 \times 96 \).

**Layer 3: Convolutional Layer**

This layer has 256 kernels, each with size \( 5 \times 5 \times 96 \). We applied a stride of 4 and a padding of 2. We also used ReLU activations for this layer. The output feature map dimension is \( 7 \times 7 \times 256 \).

**Layer 4: Pooling Layer**

The max pooling layer is identical to Layer 2, with a kernel size of \( 3 \times 3 \) and a stride of 2. The output feature map has size \( 3 \times 3 \times 256 \).

**Layer 5: Convolutional Layer**

This layer has 384 kernels, each with size \( 3 \times 3 \times 256 \). There is a stride of 1 and padding 1. ReLU activations are again used in this layer. The output feature map has size \( 3 \times 3 \times 384 \).
Layer 6: Fully-connected Layer

The fully connected layer has 1024 output units. It is fully connected, so the connections are dense and the output is the full matrix multiplication between the weights and the input. ReLU activations are applied to the results.

Layer 7: Fully-connected Layer

This fully connected layer has 256 output units. Again, its connections to the previous layer’s output are dense.

Layer 8: Fully-connected Layer

This final layer has a single unit. This is so we can assign the image a real-valued score. Since there is no normalization here, the value of the score does not have any meaning, and it only takes on meaning when it is compared to the predicted score of another image. This scoring is a relative scoring, and allows us to predict which image has a higher true score.

We will call the CNN described above CNN_A. We also had some variations on CNN_A. In particular, we had another model, CNN_BN, with the same architecture, but with batch normalization layers inserted after the first 2 convolutional layers, so that the outputs of the convolutional layers were normalized. Batch normalization is a very recently introduced technique that performs normalization over each training mini-batch, and has been shown to prevent overfitting, speed up training, and improve accuracy rates overall (Ioffe et al., 2015). The advantage of batch normalization is that instead of just performing normalization on the input before it enters the network, normalization can take place at various stages as the forward-propagation is taking
place and becomes a part of the model architecture itself.

In addition to CNN_A and CNN_BN, we also created another variation, CNN_CN. CNN_CN has the same layers as CNN_BN, but with different kernel sizes and neurons. CNN_CN was partly inspired by VGGNet, the runner-up in ILSVRC 2014 (Simonyan et al., 2015). Although we did not emulate the depth of their network due to memory costs, we did follow their example of using smaller convolutional kernels and shorter strides. Below, we show describe only the differences:

**Layer 1: Convolutional Layer**

Layer 1 still has 96 kernels, but now, each has a size of $3 \times 3 \times 3$ and stride 2. The output feature map has dimension $113 \times 113 \times 96$.

**Layer 2: Pooling Layer**

The output feature map has dimension $56 \times 56 \times 96$. Notice that this is significantly larger than the output feature map of CNN_A and CNN_BN.

**Layer 3: Convolutional Layer**

This layer still has 256 kernels, but now, each kernel has size $3 \times 3 \times 96$ and stride 2. The output feature map size is $29 \times 29 \times 256$.

**Layer 4: Pooling Layer**

The output feature map has dimension $14 \times 14 \times 256$.

**Layer 5: Convolutional Layer**

The output feature map has dimension $14 \times 14 \times 384$. 
The remaining architecture of CNN_CN is exactly the same as CNN_BN (the variation of CNN_A with batch normalization).

Our final model architecture is a siamese network, constructed with identical copies of either CNN_A, CNN_BN, or CNN_CN. An example can be seen in Figure 4.6.

4.3 Training the Networks

We used Caffe, a deep neural network framework, to train the networks (Jia et al., 2014). Caffe allows us to train our networks on either the CPU or the GPU. Being able to train on the GPU is what makes deep learning possible. We trained our networks on NVIDIA Tesla K40c, which has 2880 cores with 12GB memory.

The objective function of our networks is the ranking loss function described in section 4.1.3. The gradients of this loss were computed during backpropagation and used to update the weights for stochastic gradient descent (SGD). We trained our networks using ADAGRAD and ADADELTA, both of which are variants to improve stochastic gradient descent. ADAGRAD was developed first, by (Duchi et al., 2011). ADAGRAD updates each parameter separately. In ordinary SGD, the update rule for the $i$-th parameter is:

$$W_{t+1,i} = W_{t,i} - \alpha \cdot \nabla \theta L(\theta_t)_t,$$

where $L$ is our ranking loss function. ADAGRAD updates $\alpha$ for each $t$, for each $i$, using all the past gradients of $W_t$. ADADELTA (Matthew D. Zeiler, 2012) improves on ADAGRAD by also updating each parameter separately, but instead of using all
the past gradients of $W_t$, it keeps a running average. This makes ADADELTA much cheaper, and even more robust to learning rate initialization, as its update rules do not even depend on the base learning rate.

To train our networks, we used a batch size of 32 to reduce the memory usage and training time. We used the data pairs described in Chapter 3.4. We initialized our weights from a Gaussian distribution with 0 mean. Each layer was initialized with a different standard deviation, according to the formula $\text{stdev} = \frac{1}{\sqrt{\text{input size}}}$ (Glorot et al., 2010). As an example, this means that the first convolutional layer, whose input is the image data with dimensions 227 × 227 × 3, would have a weight initialization with standard deviation 0.003. All biases were initialized to 0. We trained each network for a total of 100,000 iterations, with validation run on the dataset every 500 iterations.

We performed another variation by training some of our networks on data pairs formed by pairing up all the people in the dataset, without the requirement of each pair being only one person. Everything else about the network architecture and learning rules were kept the same. The motivation for doing this is that training on all pairs of people’s photographs would serve the purpose of giving us some insight as to what our network might be learning.

### 4.4 Hyperparameter Optimization

We want to choose model and solver parameters that gives us the highest accuracy on the validation set. Instead of tuning the models by hand, we decided to explore the parameter space more systematically. As it is often impractical if not impossible to obtain meaningful derivatives of these hyperparameters, we opted to use
a randomized black-box optimization technique instead of the classical methods based on gradient-descent or Newton’s method. In randomized black-box optimization, we seek to minimize an objective function

\[ f : \mathbb{R}^n \rightarrow \mathbb{R} \]

by repeating the following steps:

- Pick samples of input values \( \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_\lambda \in \mathbb{R}^n \) based on some distribution \( D \)
- Evaluate \( f \) on \( \bar{x}_1, \bar{x}_2, \cdots, \bar{x}_\lambda \)
- Update the input sampling distribution \( D \) based on \( f(\bar{x}_1), f(\bar{x}_2), \cdots, f(\bar{x}_\lambda) \)

The sampling and update steps are constructed so that the function values tend to decrease on every pass. The specific sample and update scheme we will use is the covariance matrix adaptation evolution strategy (CMA-ES) introduced by (Hansen et al., 1996).

**Sampling**

The trial solutions in CMA-ES are sampled from a multivariate Gaussian distribution

\[ \bar{x}^{(k+1)}_i \leftarrow \bar{m}^{(k)} + \sigma^{(k)} \mathcal{N}(0, \bar{C}^{(k)}) \]  

(4.1)

with mean \( \bar{m} \), covariance matrix \( \bar{C} \), and overall standard deviation \( \sigma \), where the superscripts in parentheses denote the current generation.
Updating

Given $\lambda$ input-output pairs, CMA-ES’s update scheme picks $\mu$ number of pairs with the lowest output. The mean of the distribution is updated with as a weighted average of the $\mu$ inputs

$$m_{(k+1)} = m_{(k)} + c_m \sum_{i=1}^{\mu} w_i (x^{(g+1)}_{j(i)} - m^{(g)}) , \tag{4.2}$$

where $j(i)$ is the index of the $i$-th smallest output, and the weights $w_i$’s must satisfy

$$\sum_{i=1}^{\mu} w_i = 1 \quad w_1 \geq w_2 \geq \cdots \geq w_{\mu} \geq 0 .$$

The update equation for the covariance matrix $\bar{C}$ is a more complicated.

$$\bar{C}_{(k+1)} = (1 - c_1 - c_\mu) \bar{C}^{(k)} + c_1 \bar{p}_c^{(k+1)} (\bar{p}_c^{(k+1)})^\top + c_\mu \sum_{i=1}^{\mu} w_i y^{(k+1)}_{j(i)} (y^{(k+1)}_{j(i)})^\top \tag{4.3}$$

where

- $c_1, c_\mu \in (0, 1]$ are two learning rates
- $\bar{p}_c$ is the evolution path, which accounts for the direction of change from the previous generation
- $\bar{y}_i$ is defined to be the function outputs $f(\bar{x}_i)$

The covariance matrix is updated to increase the likelihood that solutions near the previous $\mu$ solutions with the lowest function values are sampled in the new generation. The exact values we used for CMAE-ES parameters include convolution stride,
kernel size, number of fully-connected outputs, etc. These values come from the configurations for model CNN_A.
Figure 4.6: An example of our siamese network architecture with ranking loss
Chapter 5

Results and Discussion

In this chapter, we will present and discuss the results of the different networks. First, we will discuss and compare the performance of each network architecture and the learning rules. We will then analyze the results and see if there is any insight to be gained.

5.1 Network Performance

In the sections below, we will discuss the accuracy of CNN_A, CNN_BN, and CNN_CN. In the subsequent sections, when we refer to accuracy, we are referring to the percentage of pairs of the test set or a subset of the test set for which our network predicts the higher-scoring image correctly. We compare the accuracy the networks achieves with chance, which is 50% on this dataset. We will also refer to two test sets - test_same and test_different. The dataset test_same consists of 4,544 pairs of images, where each pair is formed between the same person’s pictures only. The dataset test_mixed consists of 30,000 pairs of images, where each pair can consist of...
images of the same person, or of two different people.

5.1.1 CNN_A

Training on Mixed-people Pairs

We trained this network on pairs including combinations of different people. We used ADADELTA with $\delta = 1e - 6$, momentum= 0.95 and weight-decay= 0.0005. Although we attempted to fine-tune the network using the hyperparameter optimization techniques described in the previous chapter, we were not able to get satisfactory results due to time constraints, and will have to leave this for future investigation. We trained this network for 25,000 iterations, at which point validation accuracy began to drop. We took this to be an indication that the network was starting to overfit, so we stopped training at this point. We tested the resulting network on the two test sets. On test_same, the network achieved 60.27% accuracy and on test_mixed, it achieved 59.49% accuracy. It was interesting to see that the accuracy achieved on test_mixed and test_same are not very different.

5.1.2 CNN_BN

Training on Same-person Pairs

For the same-person pairs training, we only used pairs that consist of images of the same person. We trained CNN_BN using ADADELTA, with the same parameters described above. We trained this for a total of 100,000 iterations, but did not see much improvement over time. The network achieved an accuracy of 59.18% accuracy on test_same, and 58.04% on test_diff.
We then trained the same network using ADAGRAD, with a base learning rate of 0.001 and a weight-decay of 0.0005. Although ADAGRAD is adaptive, a base learning rate still had to be given. Its accuracy on test_same increased very gradually over time, and achieved a final 58.21% after 130,000 iterations.

**Training on Mixed-people Pairs**

We trained in the exact same manner as above, but used mixed-people pairs as input data. We first tried ADADELTA and trained for a total of 100,000 iterations. The network achieves an accuracy of 58.90% on test_mixed pairs, and 58.16% on test_same pairs. Using ADAGRAD, we achieved slightly worse results, with 57.33% on test_same.

### 5.1.3 CNN_CNN

**Training on Same-person Pairs**

We trained the network using ADADELTA for 100,000 iterations, although there was very little improvement after the first 5,000 iterations. Interestingly enough, validation accuracy did not drop over time either, so there was no overfitting occurring. We were able to achieve 60.43% on test_same and 60.29% on test_mixed.

**Training on Mixed-people Pairs**

Using ADADELTA and 100,000 iterations again, we found similar results to those when training on same-person pairs. There was very little change in the accuracy rate over time, with accuracy peaking out around 20,000 iterations. Overall, we achieved an accuracy of 60.37% on test_same and 60.09% on test_mixed.
5.2 Discussion

We noticed a few things that are interesting.

1. The network achieved slightly better performance without batch normalization. However, the difference is not large enough for us to conclude whether it is of any significance. This could be something we can investigate further in the future.

2. ADADELTA and ADAGRAD gave us very similar results, with ADADELTA doing slightly better than ADAGRAD, by around 1%. ADADELTA is based off ADAGRAD and was designed to be an improvement to it, and our results seem to support this.

3. We found that CNN-CN outperformed CNN_BN by around 1%, with the only difference between the two networks being the sizes of the convolution kernels and strides. This supports the idea that smaller convolutional kernels and shorter strides are indeed better for performance. Intuitively, we can think of smaller convolutional kernels as smoothing out over smaller patches, which in turn retains more information. This additional information is probably what contributes to the improvement in performance.

4. We found that training with just the same-person pairs versus training with mixed-people pairs did not result in significantly different accuracy results. This could mean that our network has not been fully fine-tuned for the specific task of choosing the most attractive selfie. Instead, it might be learning some general trait of what an attractive or good photograph is.
5. Similar to the point above, we found that testing on same-person pairs and testing on mixed-people pairs gave similar results, regardless of whether the networks had been trained on same-person or mixed-people pairs. Although there was some variation, neither same-person nor mixed-people pairs had consistently higher accuracy rates. Again, we think this might be a result of the network learning what is an attractive quality in general, for example, a smiling face is more attractive than an unsmiling face.

6. We examined some of the models rankings visually; two examples are shown in Figure 5.1 and 5.2. From the images, it seems as if our model predicts well on the really bad pictures, and on the really good pictures. In certain cases, it does seem to assign a low score to an image with many votes, but we can see that in many cases, the model ranks correctly the images that were given 0 votes or over 6 votes. This seems to match the statistics of the human votes too - people tend to agree on what the best picture and the worst picture is, but there is much more variation on the intermediate images. However, it is difficult for us to make more conclusions without more human data.

Overall, our best network’s predictions are around 11% above chance and although 61% may not sound so impressive, these results are promising and indicate that there is definitely something more than subjectivity when it comes to a viewer finding one photograph more attractive than another.
(a) Votes for each image given by workers - true score assigned by our model

(b) Relative score predicted by our model. Top left is the predicted best image, bottom right is the predicted worst image

**Figure 5.1**: Ranking as decided by 9 human workers versus ranking as predicted by our model
Figure 5.2: Ranking as decided by 9 human workers versus ranking as predicted by our model
Chapter 6

Conclusion and Future Work

In this project, we designed several deep networks to predict which selfie would be the most attractive one. We have successfully trained a siamese CNN model that chooses the more attractive image from a pair correctly 61% of the time. These findings indicate to us that there is some pattern in what makes a photograph attractive, and that this pattern is learnable by machines.

In future works, we would like to investigate some other network architectures that might give us higher accuracy rates. We would like to spend more time fine-tuning the network using CMAE and other hyperparameter optimization techniques. We would also like to explore different datasets, perhaps one that is a more natural representation of selfies on the internet. It would be interesting to analyze the performance on a better dataset that is labeled by a more diverse set of workers. It would also be interesting to further explore the reasons the models achieved the same results on within-person and between-people pairs. Extending this research and improving its accuracy not only gives us a better tool for predicting facial attractiveness and for picking our best selfies, but also gives us insight into how the perception of beauty
works in general. We hope that in our future work, we will be able to make more contributions to the study of beauty.
Bibliography


Bobst, Cora and Janek S Lobmaier (2012). “Men’s preference for the ovulating female is triggered by subtle face shape differences”. In: Hormones and behavior 62.4, pp. 413–417.


Conference on Artificial Intelligence and Statistics (AISTATS) 9, pp. 249–256. ISSN: 1532-4435. DOI: 10.1.1.207.2059.


Gray, Douglas, Kai Yu, Wei Xu, and Yihong Gong (2010). “Predicting facial beauty without landmarks”. In: Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6316 LNCS.PART 6, pp. 434–447. ISSN: 0302-9743. DOI: 10.1007/978-3-642-15567-3_32.


Hayes, Andrew F. and Klaus Krippendorff (2007). “Answering the Call for a Standard Reliability Measure for Coding Data”. In: Communication Methods and Measures 1.1, pp. 77–89. ISSN: 1931-2458. DOI: 10.1080/19312450709336664.


Wang, Y., L. Zhe, X. Shen, R. Mch, G. Miller, and G. W. Cottrell (in review). “Joint Event Recognition and Image Curation”. In:


