HAIR SEGMENTATION USING HEURISTICALLY-TRAINED NEURAL NETWORKS

by

Wenzhangzhi Guo

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2016 by Wenzhangzhi Guo

Abstract

Hair Segmentation Using Heuristically-Trained Neural Networks

Wenzhangzhi Guo

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2016

We present a method for binary classification using neural networks that performs training and classification on the same data using the help of a pre-training heuristic classifier. The heuristic classifier is initially used to segment data into three clusters of high confidence positives, high confidence negatives, and low confidence sets. The high confidence sets are used to train a neural network (NN) which is then used to classify the low confidence set. Applying this method to the binary classification of hair vs. non-hair patches, we obtain a 9% performance increase using the heuristically-trained NN over the current state of the art hair segmentation method.

Acknowledgements

First and foremost, I would like to sincerely thank my supervisor Professor Parham Aarabi for his support and mentorship throughout my thesis. I am very grateful to have him as my supervisor. This thesis could not have been possible without his help and guidance.

I would also like to thank Professor Deepa Kundur, Professor Dimitris Hatzinakos, and Professor Li Qian for being members of my thesis defense committee. I appreciate all the feedback and comments regarding the thesis. At the same time, I am very thankful of them for taking their time to be a part of my thesis defense.

At last, I would like to thank ModiFace Inc. for providing the hair image database used in this thesis.

Contents

1	Introduction	1
1.1	Motivation and Objective	1
1.1.1	Hair Segmentation	1
1.1.2	Methodology	1
1.2	Contributions	3
1.3	Thesis Organization	3
2	Prior Work and Background	4
2.1	A Short Overview of Hair Segmentation Methods	4
2.2	ModiFace Heuristic-based Hair Segmentation	5
2.2.1	Image Resizing	5
2.2.2	Facial Coordinates Detection	5
2.2.3	Fixed-location Hair Sample Collection	6
2.2.4	Adaptive Hair Sample Collection	8
2.2.5	Hair Color Model Construction	10
2.2.6	Hair Gradient Model Construction	11
2.2.7	Hair Patches Color Comparison	11
2.2.8	Hair Patches Gradient Comparison	11
2.2.9	Applying Occurrence Prior Probability Model	12
2.2.10	Binary Hair Mask Construction	12
2.2.11	Threshold Reduction Based on Neighbours	12
2.2.12	Cropping Eye Regions	13
2.2.13	Finding the Largest Connected Component	13
2.2.14	Gap Filling	13
2.2.15	Hair Mask Blurring	13
2.3	Neural Networks	14
2.4	Hair Texture Analysis	15
2.4.1	Introduction	15
2.4.2	Texture Evaluation Procedure	16
2.4.3	Texture Evaluation Result	18
2.4.4	Discussion	22
2.5	Discussion	23

3	Deep Neural Network (DNN) Based Hair Segmentation	25
3.1	Proposed Algorithm	25
3.1.1	Model Components	25
3.1.2	Using Generative Model as a Feature Extractor	27
3.1.3	Using Generative Model to Pre-train the Weights of the Neural Network	28
3.1.4	Post-processing	30
3.2	Experimental Result	30
3.2.1	Dataset	30
3.2.2	Facial Coordinates Detection	30
3.2.3	Performance Measure	30
3.2.4	Experiments for Using Generative Model as a Feature Extractor	32
3.2.5	Experiments for Using Generative Model to Pre-train the Neural Network Weights	40
3.2.6	Sample Output	41
3.3	Discussion	42
4	Heuristically-Trained Neural Network (HNN) Hair Segmentation	43
4.1	Training Set Generation	44
4.1.1	Positive Training Sample Generation	46
4.1.2	Negative Training Sample Generation	46
4.1.3	Training Set Generation	48
4.2	Experimental Results	48
4.2.1	Neural Networks Structure	48
4.2.2	Performance Measure	49
4.2.3	Hair Mask Constructed Based on Heuristically-Trained Neural Network Output . .	49
4.2.4	Combined Hair Mask	55
4.2.5	Runtime	56
4.2.6	Results Comparison	58
4.2.7	Discussion	61
5	Conclusions	62
	Bibliography	64

List of Tables

2.1	Hair Sample Locations	6
2.2	Main hair angle and standard deviation of a single hair patch	19
3.1	Confusion Matrix	32
3.2	Accuracy and over-estimation comparison on the effect of the prior probability model . .	33
3.3	Performance score comparison on the effect of the prior probability model	33
3.4	Accuracy and over-estimation for different sample ratios and patch sizes	34
3.5	Precision and recall for different sample ratios and patch sizes	34
3.6	Performance score for models with different sample ratios and patch sizes	35
3.7	F1 score for models with different sample ratios and patch sizes	35
3.8	Performance score for different number of hidden layers, each layer has 1000 hidden units	36
3.9	F1 score for different number of hidden layers, each layer has 1000 hidden units	36
3.10	Performance score for different number of hidden units per layer in a four-hidden-layers network	36
3.11	F1 score for different number of hidden units per layer in a four-hidden-layers network . .	37
3.12	Performance comparison on the effect of the generative model for a neural network with 1 hidden layer and 1000 hidden units	37
3.13	Training time for neural network with different number of hidden layers, each layer has 1000 hidden units	38
3.14	Training time for neural network with different number of hidden units, with 4 hidden layers	38
3.15	Testing time (per image) for different number of hidden layers in the neural network, each layer has 1000 hidden units	39
3.16	Testing time (per image) for different number of hidden units in the neural network, with 4 hidden layers	39
3.17	Performance comparison with and without neural network weights pre-training	40
3.18	Training time comparison for pre-training a neural network with 4 hidden layers, each layer consists of 1000 hidden units	40
4.1	Detailed performance of heuristically-trained neural network hair masks generated based on training with ground truth	49
4.2	Performance score for hair masks based on the HNN output	51
4.3	F1 score for hair masks based on the HNN output	51
4.4	Best performance comparison for HNN hair mask	53

4.5	Performance comparison for different number of training samples for HNN hair masks . .	53
4.6	Performance of multi-layer HNN	55
4.7	Best performance comparison for combined hair masks	56
4.8	Runtime breakdown for heuristically-trained neural network for a single image	56
4.9	Performance comparison of different methods	58
4.10	Performance for Fig 4.14 comparison	58
4.11	Performance for Fig 4.15 comparison	58

List of Figures

2.1	An overview of the ModiFace method [8]	6
2.2	Sample image showing hair samples regions extracted	7
2.3	Boundaries detected in the adaptive hair sample collection	9
2.4	Hair samples collected in the adaptive hair sample collection	10
2.5	Hair occurrence prior probability model	12
2.6	Blurring comparison: left: hair mask before Gaussian smoothing; right: hair mask after Gaussian smoothing	14
2.7	Sample hair patches	18
2.8	Example hair patch	19
2.9	Histogram using gradient magnitude as bin count	19
2.10	Average standard deviation for the different patch sizes	20
2.11	Top-left: Average main hair angle color map for 4×4 patch size; Top-right: Average main hair angle color map for 8×8 patch size; Bottom: Color Legend, median angle is the median of the directions that a color is responsible for	22
2.12	Top-left: Standard deviation color map for 4×4 patch size; Top-right: Standard deviation color map for 8×8 patch size; Bottom: Color Legend, median angle is the median of the angle range that a color is responsible for	23
3.1	Multilayer neural network model architecture	28
3.2	System overview: a) Take patches from the input image b) Vectorize the patch c) Extract features with the generative model d) Classify the features with neural networks	29
3.3	Top row: Sample images from the database; Bottom row: Corresponding hair cut-out	31
3.4	Graphical illustration of accuracy and over-estimation: accuracy = d/t and over-estimation = e/t	32
3.5	Training time for neural network with different number of hidden layers, each layer has 1000 hidden units	38
3.6	Training time for neural network with different number of hidden units, with 4 hidden layers	39
3.7	Successfully segmented hair samples. Top row: input image; Bottom row: output from the proposed algorithm (shown as red pixels)	41
3.8	Sample failure cases. Top row: input image; Bottom row: output from the proposed algorithm (shown as red pixels)	42
4.1	Initially: Data is not labelled or arrange in any way	44

4.2	Step 1: We apply a heuristic rule to sort data	44
4.3	Step 2: We separate data into high-confidence regions and low-confidence regions	45
4.4	Step 3: We use high-confidence positive and negative samples to train a neural network	45
4.5	Step 4: We use the trained neural network to classify all data	46
4.6	Sample hair masks. First row: input image; Second row: big hair mask; Third row: small hair mask; Fourth row: negative samples generated with small hair mask	47
4.7	Sample outputs: First row: input image; Second row: ModiFace [8] hair mask output ; Third row: HNN hair mask output	50
4.8	Performance scores for different thresholds (γ_{thres}) for HNN hair mask	52
4.9	F1 scores for different thresholds (γ_{thres}) for HNN hair mask	52
4.10	Performance score for different rotation steps	54
4.11	F1 score for different rotation steps	54
4.12	Sample outputs: First row: input image; Second row: ModiFace hair mask output [8]; Third row: HNN hair mask output	55
4.13	Sample outputs: First row: input image; Second row: ModiFace hair mask output [8]; Third row: HNN hair mask output; Fourth row: combined hair mask output	57
4.14	First sample comparison between the ModiFace hair mask [8] and the HNN output. First column shows the result of using an aggressive ModiFace hair mask [8]; the second column shows the result of using a conservative ModiFace hair mask [8]; the third column shows the HNN output	59
4.15	Second sample comparison between the ModiFace hair mask [8] and the HNN output. First column shows the result of using an aggressive ModiFace hair mask [8]; the second column shows the result of using a conservative ModiFace hair mask [8]; the third column shows the HNN output	60

Chapter 1

Introduction

1.1 Motivation and Objective

1.1.1 Hair Segmentation

The goal of this thesis is to segment hair out from the input image. Hair segmentation is a key step in many different applications such as beauty applications, photo adjustment, and facial identification. Since hair plays an important role in the overall appearance of a person, many beauty applications focus on designing and modifying the hairstyles of individuals. In many of these cases, instead of directly operating on the individual, the desired effect will be first shown on a photo of the individual. Therefore, hair region needs to be first detected before any changes can be applied to the current hairstyle in the photo. Furthermore, in order to facilitate the hairstyle modification process, there has been increasing interest in hair-based image processing including hair synthesis [1], hairstyling [2], hair rendering [3], and hair simulation [4].

Hair segmentation is also needed in photo adjustment/editing. In photo editing, we often want to modify the background but keep the foreground object (i.e. people) unchanged. In those cases, we need to segment out the foreground object, including hair.

Segmented hair can also be used for identification purposes. Hair can facilitate gender classification since the hairstyles for males and females are generally different [5]. Additionally, hair can be used as an important basis for facial recognition tasks since the hairstyle for an individual rarely changes frequently [6].

1.1.2 Methodology

Segmenting out hair by hand can be quite tedious and time-consuming. Thus, people have developed different techniques to segment hair automatically.

The most straightforward way of automatic hair segmentation is based on heuristics since hair has very

distinct color and texture when comparing to the rest of the image. In heuristics-based methods, the first step is usually to find good candidates for the hair region (hair seeds) based on a prior probabilistic hair mask or fixed location relative to the face. The final segmentation result is obtained by matting or comparing with the hair seeds for the rest of the image [7] [8]. However, heuristic-based hair segmentation is highly dependent on the many threshold values being used (e.g. the measure of similarity for color and texture) as well as the accuracy of facial detection. Thus, heuristic-based method usually has large performance variance across different images (since it is very difficult to find a universal threshold that works for every single image).

Another class of hair segmentation techniques uses machine learning algorithms. The idea of machine-learning-based technique is instead of finding the optimal threshold by hand, we train a machine to automatically select good features and perform the hair segmentation. There are two classes of machine learning techniques: supervised learning (inferring a function from labelled training data) and unsupervised learning (inferring a function from unlabelled training data). The most popular unsupervised learning technique used in segmentation problems is clustering, where we try to put similar objects into the same group [9] [10]. On the other hand, in supervised learning, we first use labelled training data to train a classifier, and then use the trained classifier to perform the segmentation task on the whole image. In the training process, the classifier will automatically learn good features and parameters based on the training samples. There are many supervised learning algorithms being used for image segmentation, such as support vector machine [11] [12], decision trees [13] [14], and neural networks [15] [16]. In this thesis, we are going to focus on hair segmentation with neural networks since they have very flexible structures and are not very difficult to train.

The use of neural networks can be dated back to 1958 when Rosenblatt first introduced the perceptron [17]. Then the introduction of the back-propagation algorithm made it feasible to train a non-linear neural network [18]. However, training a neural network with multiple hidden layers (deep learning) was very difficult until the introduction of pre-training, where people can pre-train the neural network to achieve much better performance [19]. Deep learning has been successfully applied by researchers in several different fields for tasks such as image classification [16] [20], speech recognition [21] [22], and language parsing [23] [24].

In this thesis, we want to focus specifically on image segmentation with neural networks, especially deep neural networks. Generally, deep neural networks refer to neural networks with more than two hidden layers [25]. Compared to shallow architectures (neural networks with one or two hidden layers), it is presumed that deep neural networks process information in a way that is closer to the human brain, especially the human visual system [26]. With a deep architecture, the inputs in a neural network can be represented in a hierarchical way, with multiple levels of abstraction [25]. This deep architecture is particularly useful for hair segmentation since the appearance of hair can be quite different across images. A deep neural network models the hair appearance through different levels of abstraction, where each abstraction consists of activating different hidden units in a hidden layer [25].

Throughout this thesis, we are going to explore neural networks with both shallow and deep architectures and evaluate their performance on hair segmentation. We will implement and compare two different neural network systems. The first system aims to learn from hair and non-hair patches taken from many different images. The trained neural network can then be used to detect hair in any input image.

The second system first segments each input image into three clusters of high confidence positives, high confidence negatives, and low confidence sets with a heuristic-based classifier. The high confidence sets are used to train a neural network (NN) which is then used to classify the low confidence set.

1.2 Contributions

This thesis has many contributions on neural network architectures for hair segmentation. The top two contributions are the followings:

1. Design and implementation of a heuristically-trained neural network

A heuristic classifier is initially used to segment data into three clusters of high confidence positives, high confidence negatives, and low confidence sets. The high confidence sets are used to train a neural network (NN) which is then used to classify the low confidence set. We have shown experimentally its benefits and compare its performance with the state of the art hair segmentation technique.

2. Hair patch texture analysis

We evaluate the distribution of gradient directions in hair patches, and explore the relation between proximity to the face and the angle of the gradients. We quantitatively evaluate the role of texture in hair patches, which illustrates what can be learned and applied by machine learning systems for texture-based hair detection.

1.3 Thesis Organization

This thesis is organized into 5 chapters. Chapter 1 motivates the thesis and outline the structure of the thesis. Chapter 2 presents several prior work done on hair segmentation, as well as the hair patch texture analysis. Chapter 3 presents a universal neural network that can be applied to segment hair for any input image. Chapter 4 presents the heuristically-trained neural network whose training samples are selected using clustering based on human-defined heuristics. Lastly, Chapter 5 summarizes the main findings of the thesis and lists the future work for this thesis.

Chapter 2

Prior Work and Background

2.1 A Short Overview of Hair Segmentation Methods

Since the appearance of hair can vary between different people based on their gender, age, ethnicity, and the surrounding environment, automatic hair segmentation is challenging in general. Automatic hair segmentation methods can be roughly divided into four groups. The first group of prior work performs hair segmentation by first extracting several hair seeds and then passing them into a pre-trained classifier. For example, in [27], the author first generated hair seed pixels by combining the hair occurrence prior probability with a generic hair color model to construct a Bayesian model. Then the hair seeds generated were passed into a second Bayesian model constructed by combining a pre-trained Gaussian model with the hair occurrence prior probability. This second Bayesian model was responsible for classifying the whole input image. However, this model will have poor performance when we have low contrast between the hair color and background color since this classification method is primarily based on color information. In [28], the author first used Active Segmentation with Fixation (ASF) to find a good candidate hair region, and then graph-Cut was used to refine this region. Next, Bayesian method was applied to find hair seeds in the image, and a SVM classifier and graph-Cut were used to further classify the whole image.

The second group of prior work is based on the optimization of a rough estimate of the hair mask. In [29], the author first constructed a Coarse Hair Probability Map (Coarse HPM) based on the color and location prior and then refined the Coarse HPM based on the similarity of the current Coarse HPM comparing to Coarse HPMs of images in the training set (since hair with similar Coarse HPM should be segmented similarly). In [30], a mixture model was first built based on the hair color and location information to get a rough estimate of the face, hair, and background regions. Then, graph-Cut or Loopy Belief Propagation algorithm was used to optimize the resulting Markov network for the hair classification task.

The third group of prior work is based on a mixture of different techniques. For example, in [31], the author used a compositional model consisted of techniques such as Divide-and-Conquer and Dual-Level Conditional Random Fields to perform hair segmentation.

Lastly, several other prior work is based on heuristics where the first step is usually to find good candidates for the hair region (hair seeds) based on a prior probabilistic hair mask or fixed location relative to the face. The final segmentation result is obtained by matting or comparing with the hair seeds for the rest of the image. For example, in [32], the author first found the upper hair part by minimizing an energy which used active shape and active contour. Then a pixel-wise optimization technique was used to further improve the segmentation result around the upper hair part. Another prior approach was to perform matting on a binary mask created by frequential and color analysis of the hair location [7]. However, this method is highly sensitive to the accuracy of the face detection, and the hair model is built based on a head area defined based on experience. In the section below, we will describe another heuristics-based hair segmentation method in details since it will be used to select training samples in Chapter 4.

2.2 ModiFace Heuristic-based Hair Segmentation

As mentioned in Section 2.1, the heuristic-based method performs hair segmentation mainly based on color and texture information of each input image. The ModiFace heuristic-based hair segmentation method is implemented based on [8]. The ModiFace method [8] first tries to build a model for hair color and for hair gradient from hair samples obtained in the input image based on locational information and then utilizes these models to classify the rest of the image. The performance of the ModiFace method [8] is highly dependent on the threshold. In other words, the aggressiveness of the hair mask can be adjusted by adjusting the threshold. An overview of the ModiFace method [8] is shown in Fig 2.1.

2.2.1 Image Resizing

Before performing any operations on the input image, we first resize the image so that the face widths of all the input images are the same. We usually do not want to make the original image bigger after the resizing as it adds no extra information and also increases the processing time. In this case, the resized face width is 100 pixels since most face widths in the input images are greater than 100 pixels and it leaves the image with enough information for further processing. This step turns all input images into similar scale and thus makes it easier to find the optimal threshold values that achieves good performance for most images (since the pixel color similarity threshold for a high resolution image and a low resolution image is quite different).

2.2.2 Facial Coordinates Detection

In order to find good candidates of hair patches used to build the hair models, we first need to detect the coordinates of the face, eyes, and mouth. We used the algorithm described in [33] to detect facial coordinates in each input image. The face bounding box is useful when applying the prior probability mask and extracting the hair samples. The eyes and mouth coordinates are used to find the skin color (they need to be cropped out when finding the average skin color).

Figure 2.1: An overview of the ModiFace method [8]

2.2.3 Fixed-location Hair Sample Collection

After getting the facial coordinates, we first try to get hair samples for the hair model based on fixed locations relative to the face. We selected three locations relative to the face to take hair samples (since most female hairstyles should have hair appear in those locations) as shown in Table 2.1 (L = left face coordinate; R = right face coordinate; T = top face coordinate; B = bottom face coordinate; W = face width; H = face height) and a sample of the hair samples extracted (in red rectangles) is shown in Fig 2.2.

Table 2.1: Hair Sample Locations

hair sample location	left boundary	upper boundary	right boundary	lower boundary
top	$L + 17\% \text{ of } W$	$T - 20\% \text{ of } H$	$R - 17\% \text{ of } W$	$B - 115\% \text{ of } H$
left	$L - 15\% \text{ of } W$	$T + 15\% \text{ of } H$	$R - 105\% \text{ of } W$	$B - 60\% \text{ of } H$
right	$L + 105\% \text{ of } W$	$T + 15\% \text{ of } H$	$R + 15\% \text{ of } W$	$B - 60\% \text{ of } H$

However, in some cases, the rectangles we extracted using the fixed-location sample collection might not contain hair. Therefore, we compare the hair samples we collected from the three rectangular regions to see how similar they are. In this case, we compare the color of the regions based on the similarity in the color histogram using the following steps:

1. Construct a 3D color histogram with 10 bins for each color channel for each region.
2. Normalize the 3D color histogram by dividing each bin number by the sum of all the bin numbers

Figure 2.2: Sample image showing hair samples regions extracted

for each region.

3. Go through each bin in the 3D color histogram of the two regions under comparison, and take the minimum of the two bin numbers to construct a minimum color histogram.
4. Take the sum of all the bin numbers in the minimum color histogram. This sum is the similarity measure of the two regions.

We compare two of the three hair sample regions at one time, and get the histogram similarity score. If this score is higher than a threshold, then the two regions are similar. If all three regions are similar to each other, we take all three regions in as the model hair samples. If two out of three regions are similar, we take these two regions in as the model hair samples. If all three regions are not similar at all, then the fixed-location hair sample collection fails to collect good samples, we move on to the adaptive hair sample collection.

2.2.4 Adaptive Hair Sample Collection

Adaptive hair sample collection takes more time to run than the fixed-location hair sample collection. Thus, we only use this method if the fixed-location hair sample collection fails to collect good hair samples.

Skin Detection

We first crop out the eye regions and mouth region and then crop out an elliptical region that is slightly smaller than the face bounding box. We construct a 3D color histogram for this elliptical region and use it as the model for the skin color. Taking the whole face region instead of a small area (e.g. the tip of the nose) to find the skin color is more beneficial since it weakens the effects of highlight or shadow at any particular region.

Forehead Detection

We take a small rectangular region above the face bounding box, between the eyes and construct a 3D histogram for this region. We compare it with the skin color histogram and see if the histogram similarity score is above a threshold. If so, then we have successfully find part of the forehead region. If not, we move the rectangular box to the left and to the right, and compare it with the skin color model. If we still cannot find the forehead region, we move the rectangular box down and repeat the same procedure. We stop the forehead detection procedure either after we have found the forehead region, or we have reached the top of the face bounding box.

Bang Detection

If we stop the forehead detection procedure due to the fact that we hit the top of the face bounding box, then the hairstyle in the input image has bangs. Since we have not found the forehead region successfully, it is not possible to perform the adaptive hair sample collection procedure described in the next section ("Hair Sample Collection"). In this case, we only take a rectangular region at a fixed distance above the top of the face bounding box as the model hair sample.

Hair Sample Collection

In this section, since we cannot construct a histogram based on only one pixel value, the color comparison is based on ΔE score [34]. The ΔE score is calculated by first converting the two RGB color into the $L^*a^*b^*$ color space and then calculate the Euclidean distance of the two colors (ΔE score) using the following equation [34]:

$$\Delta E_{Lab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2} \quad (2.1)$$

If the input image does not have bangs, then we have successfully detected the forehead region (an example is the blue region shown in Fig 2.3). We first need to find out how many times we can move up until we hit the hair region above the forehead. For each pixel in the top boundary of the forehead region, we want to expand it upward and compare the current pixel with all pixels within 3 pixels vertically above it (using the ΔE score). We stop expanding it upwards if we see the ΔE score is above a threshold when we compare the current pixel with all 3 pixels above it, and set the current pixel as the top hair boundary of the current column (red dots in Fig 2.3). Then the upper boundary of sample collection is the average of the top hair boundaries of all columns in the forehead patch (we take the average of the top hair boundaries instead of taking samples for each column individually based on the top hair boundary for that column since detection based on a single pixel is more prone to errors caused by shadows and highlights). We then take a longer rectangular region slightly above the upper boundary of sample collection as the top hair sample region (red block in Fig 2.4).

Figure 2.3: Boundaries detected in the adaptive hair sample collection

In the meanwhile, we also collect hair samples on both sides of the face by expanding the forehead regions. For every pixel on the left boundary of the forehead region, we expand it leftward and compare the current pixel with all pixels within 3 pixels on the left of it (using the ΔE score). We stop expanding it leftwards if we see the ΔE score is above a threshold when we compare the current pixel with all 3

Figure 2.4: Hair samples collected in the adaptive hair sample collection

pixels on the left of it, and set the current pixel as the left hair boundary of the current row (green dots in Fig 2.3). Then the left boundary of sample collection is the average of the left hair boundaries of all rows in the forehead patch. We then take a rectangular region slightly on the left of the left boundary of sample collection as one of the left hair sample regions (green blocks in Fig 2.4). We repeat this procedure when we expand rightwards, and the right hair boundary of the current row is shown as purple dots in Fig 2.3 and the right hair sample regions are shown as purple blocks in Fig 2.4. After getting all the left hair region and right hair region for the current forehead region, we move the forehead region upwards and repeat this procedure until the top boundary of the forehead region hits the upper boundary of sample collection.

2.2.5 Hair Color Model Construction

The goal of this step is to build a color model based on the hair samples collected, so that we can compare all the hair patches in the input image with it later on to decide if they are hair patches or not. Since each pixel color lies in a 3D space (R, G, B color space) and there are slight color changes across different

pixels in a patch, the average color cannot represent the color information of a patch very accurately (since many patches with completely different colors can result in the same average color). Instead, the hair color model is built in terms of a color histogram as it reflects the color information along with how color changes in a hair patch. Thus, after getting the sample hair regions, we build a normalized 3D color histogram with 10 bins for each color channel for each region, adding the histograms together and then normalize again to build hair color model of the input image.

2.2.6 Hair Gradient Model Construction

Hair texture is best described by gradient of a hair patch. Even though hair gradient direction follows a certain pattern, we have not found a good way to use this information yet. Thus, in this case, we only construct the hair gradient model based on the gradient magnitude of a patch. After getting the sample hair regions, we calculate the gradient magnitudes of the sample hair regions with the Sobel operator [35] and take the average of all gradient magnitudes as the hair gradient model of the input image.

2.2.7 Hair Patches Color Comparison

The goal of this step is to compare each hair patch in the input image with the hair color model and then get a color similarity score for each patch. For each input image, we divide the resized input image into non-overlapping patches of fixed size. For each hair patch, we construct a 3D color histogram with 10 bins for each color channel. We compare the 3D color histogram for the patch with the hair color model to get the color similarity score (sum of all the bin numbers in the minimum color histogram as mentioned in Section 2.2.3) for the current patch.

2.2.8 Hair Patches Gradient Comparison

Similar to the hair color comparison, the hair patches gradient comparison is also done for each non-overlapping hair patch in the input image. The hair gradient magnitude can have values ranging from less than 1 to greater than 100. Even for hair patches in the same image, the gradient magnitude can have quite different values at different locations. Therefore, it is very difficult to match gradient magnitude of a hair patch with the gradient magnitude model exactly.

We first tried to normalize the gradient magnitude score based on the exponential difference of the two gradient magnitude values. However, this normalization makes the gradient score very small for almost all hair patches, making it difficult to find a suitable threshold later on.

Instead, we only set a loose bound for the gradient magnitude matching criteria: we set the gradient similarity score of a patch to 0 if its average gradient magnitude is much larger or smaller than the model gradient magnitude value (based on a threshold value). Otherwise, we just use the average gradient magnitude value of the input patch as the gradient similarity score.

2.2.9 Applying Occurrence Prior Probability Model

The occurrence prior probability model gives us the prior probability of a pixel location being hair based on prior knowledge. The prior model is produced by first re-sizing the face bounding box of each pre-cut hair mask to the following dimensions: left-x = 200th pixel, top-y = 200th pixel, right-x = 300th pixel, bottom-y = 333th pixel. Then the re-sized pre-cut masks are super-composed together and normalized to represent the probability of a certain location being hair. A total of 115 hair masks taken from the ModiFace database were used to produce the hair prior probability model in this thesis. An example of the hair occurrence prior probability model is shown in Fig 2.5.

Figure 2.5: Hair occurrence prior probability model

2.2.10 Binary Hair Mask Construction

In this step, we construct a binary hair mask based on the color, texture, and prior probability information we obtained in the previous steps. After getting the color similarity score, the gradient similarity score and the prior probability, we multiply them together to get the total score of a patch being hair. We then threshold this total score to turn the hair mask into a binary mask (0 being non-hair and 1 being hair). We choose to multiply the different scores instead of adding them together since the hair gradient score is usually much larger than the other two scores and adding them together can cause false regions to form.

2.2.11 Threshold Reduction Based on Neighbours

In this step, we take neighbouring effect into account, aiming to find more hair pixels. In general, a patch who has a couple neighbours being identified as hair patches has a high probability of being a

hair patch (since hair regions are usually continuous). Therefore, we reduce the threshold value we use to turn the hair mask into a binary mask by half for those patches that have more than n neighbours being identified as hair, where n is a threshold we set based on experiments.

2.2.12 Cropping Eye Regions

In many cases, eyebrows can be easily mistaken as hair since they have similar color and texture as hair. At the same time, we already know where eyes are located in the image from the eye coordinates detection. Therefore, we can simply crop out a slightly larger area around the eye region in the hair mask to reduce the detection error. The cropped out eye regions are also useful in Section 2.2.14.

2.2.13 Finding the Largest Connected Component

Another valid assumption to make about the detected hair region is that it should be the largest region in the image. The hair region should be much larger than other isolated regions (e.g. clothes, background). However, in some cases, we might end up with two or more non-connected hair regions. Therefore, we save the largest connected region and any other region that is bigger than 80% of the largest connected region. All other isolated regions are discarded.

2.2.14 Gap Filling

The gap filling step aims to fill up "holes" in the detected hair regions. We apply a "water-flow" algorithm to fill in the gaps. We first start from one edge of the image and expand toward the center. For every pixel on that edge, we keep expanding until we hit a hair pixel and we label everything on its path as non-hair. We repeat the same procedure for all 4 edges of the input image. Additionally, sometimes we could detect some clothes as hair and form a closed boundary around the face. In this case, this "water-flow" technique from the 4 edges will never hit the face region and face would be considered as hair. To solve this problem, we apply "water-flow" from the 4 edges of the eye regions as well.

2.2.15 Hair Mask Blurring

The very last step is to apply Gaussian blurring to the binary hair mask. The Gaussian filter helps to smooth the hair mask and slightly change the detected region based on neighbours [36]. The "neighbouring effect" increases the detection accuracy in most cases since hair pixels tend to stay close to each other (most changes would occur on the boundary of the detected hair region). A sample comparison of the hair mask before and after the blurring is shown in Fig 2.6.

After blurring, the final hair mask is also resized into the original size of the input image.

Figure 2.6: Blurring comparison: left: hair mask before Gaussian smoothing; right: hair mask after Gaussian smoothing

2.3 Neural Networks

Neural network is one of the most popular machine learning techniques. It has been used in many different areas such as image classification [16] [20], speech recognition[21] [22], and language parsing [23] [24]. Even though we did not find any prior work of using neural networks for hair segmentation purposes, neural networks have been widely used in many other image segmentation and classification problems. Image segmentation with neural networks can be categorized into two groups: image segmentation based on pixel data and image segmentation based on features [37].

Pixel-data-based segmentation segments images directly from pixel or voxel data [37]. For example, in [38], the authors trained a multilayer feedforward neural network to localize and segment text from images and videos at fixed scale. Furthermore, in [39], the authors used a hierarchical self-organizing map to perform multiscale image segmentation. In this case, the output of one self-organizing map was used as the input to the next self-organizing map. The hierarchical structure of the self-organizing map overcame the limitation of the single-layer self-organizing map by producing various levels of abstraction in the feature space. In addition, instead of using only a single neural network, researchers have also tried to combine different neural networks together using a hierarchical approach. In such a system, one or more neural networks are used to extract different features and the outputs of these neural

networks are fed into another classifier for final segmentation. For example, in [40], the authors used a hybrid neural network to perform segmentation and classification of multispectral magnetic resonance (MR) images. They first used a Kohonen self-organizing neural network for segmentation and then a multilayer feedforward neural network for classification. In [41], the authors first fed region proposals into a convolution neural network for feature extraction. The features were then passed into a SVM for classification and segmentation.

On the other hand, instead of feeding raw pixels into the neural network, feature-based segmentation provides neural network with pre-computed feature vectors. For example, in [42], the authors first selected eight features that characterised the local texture in the neighbourhood of a pixel. Then they used these features to train a neural network to segment radiographic images. In [43], the authors first extracted gradient features from candidate regions. Those intensity gradients were then passed into a trained neural network for pedestrian detection and segmentation.

2.4 Hair Texture Analysis

In this section, we will explore the role of texture in hair patches, with a primary motivation of understanding what can be learned and applied by machine learning systems for texture-based hair detection. We evaluate the distribution of gradient directions in hair patches, and explore the relation between proximity to the face and the angle of the gradients. The analysis in this section will be published in [44].

2.4.1 Introduction

Hair texture information is obtained by calculating the gradient of the hair regions. We obtain two numbers for each gradient calculation: gradient magnitude and gradient direction. The gradient magnitude indicates how strong the texture changes and the gradient direction indicates where the change orientates towards. The gradient magnitude of hair regions can vary largely between different images depending on the resolution and hair style of the image. On the other hand, there are far less variations in the gradient directions at the same location across images. In this chapter, we construct histogram of hair angles (gradient direction rotated by 90 degrees) and use the gradient magnitude information as the bin count for each corresponding hair angle in the histogram.

Texture analysis has been used to solve problems in various fields such as medical diagnosis [45] [46] and remote sensing [47] [48]. Additionally, we found several related work that utilized gradient information to synthesize or model hair. For example, in [49], the authors applied line integral convolution (LIC) to a vector field representing the directions of hair strands to synthesize textures that are similar to human hair. In [1], the authors proposed a generative model with hair color, orientation and vector field for hair sketching. Lastly, in [50], the authors compared the histogram of the orientations of input patches taken from a hair image with patches in the database, aiming to represent the input patches with simpler line models to facilitate the later 3D re-construction of the inputs.

2.4.2 Texture Evaluation Procedure

The goal of this section is to evaluate the textural patterns in human hair in terms of the direction and distribution of hair flow. To find the direction of the hair flow in a hair patch, we first need to calculate the gradient direction and then turn the gradient direction by 90 degrees.

Calculation of the Direction of Hair Flow (Hair Angle)

There are many ways to calculate the gradient of an image patch. In this section, we used a Sobel operator to estimate the gradient of the input patch due to its simplicity. A Sobel operator consists of two 3×3 kernels is used to estimate the gradient magnitude and direction of a hair patch [35]. Given a grayscale hair patch I , the gradient magnitude and hair angle of the patch at location (x, y) can be calculated using the following steps [35]:

1. Calculate x-direction derivatives:

$$G_x(x, y) = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I(x : x + 2, y : y + 2) \quad (2.2)$$

where $x, y \in [1, patch_size - 2]$

2. Calculate y-direction derivatives:

$$G_y(x, y) = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I(x : x + 2, y : y + 2) \quad (2.3)$$

where $x, y \in [1, patch_size - 2]$

3. Calculate gradient magnitude:

$$|m(x, y)| = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \quad (2.4)$$

4. Calculate gradient direction (positive direction pointing to the right and increasing counter-clockwisely):

$$\alpha(x, y) = \text{atan2}(G_y(x, y), G_x(x, y)), \alpha \in [-180, 180] \quad (2.5)$$

5. Calculate hair angle (adding 90 degrees to convert gradient direction to hair angle; mod 180 degrees)

to keep the hair angle between 0 and 180 degrees):

$$\theta(x, y) = (\alpha(x, y) + 90) \mod 180 \quad (2.6)$$

where $*$ is a two-dimensional convolution operator, $|m(x, y)|$ is the gradient magnitude, and $\theta(x, y)$ is the hair angle in degrees.

Hair Angle Histogram Construction

After getting the hair angles and gradient magnitudes in a hair patch, we construct a histogram of hair angles. The histogram has 180 bins, from 0 to 179 degrees (any angle between 179.5 and 180 falls into the 0 degrees bin). The bin count in this case represents the energy at a particular angle in the histogram. We experiment with four different weighting functions when constructing the histogram:

For each hair angle $\theta(x, y)$ with corresponding gradient magnitude $|m(x, y)|$ in a hair patch, we add the following weight to the bin responsible for $\theta(x, y)$ in the histogram:

$$\begin{cases} w_1(|m(x, y)|) = 1 \\ w_2(|m(x, y)|) = |m(x, y)| \\ w_3(|m(x, y)|) = \sqrt{|m(x, y)|} \\ w_4(|m(x, y)|) = \log(|m(x, y)| + 1) \end{cases} \quad (2.7)$$

Calculation of the Main Hair Angle

In this section, we use the main hair angle instead of the mean hair angle to represent the direction of the hair flow in a patch. The mean hair angle is the arithmetic mean of all the hair angles in a patch. The mean hair angle can represent the direction of the hair flow in most cases, except when the hair angles are around the wrapping point (180 degrees). In this case, the mean hair angle could lay around 90 degrees, which is not accurate in this case. On the other hand, the main hair angle represents the direction with the maximum energy and can always reflect the hair direction in a patch accurately. It can be calculated with the following equation:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{i=\theta-\phi}^{\theta+\phi} h(i) \quad (2.8)$$

where $h(i)$ is the bin count at angle i in a histogram of hair angles, ϕ is the overlapping interval (it wraps around at 180 degrees), and $\hat{\theta}$ is the main hair angle.

Calculation of the Standard Deviation of Hair Angles

Standard deviation is a measure of the spread of data. The standard deviation of the hair angles in a single patch should be small since hair is very directional. To calculate the standard deviation of the hair angles, the following equation is used:

$$\sigma = \sqrt{\frac{1}{\sum_{i=0}^{179} h(i)} \sum_{i=0}^{179} h(i) \gamma^2(i, \hat{\theta})} \quad (2.9)$$

$$\text{where } \gamma(i, \hat{\theta}) = \min(|i - \hat{\theta}|, 180 - |i - \hat{\theta}|) \quad (2.10)$$

where i is the bin index in the hair angle histogram, $h(i)$ is the bin count for bin i and $\hat{\theta}$ is the main hair angle calculated in the previous section.

2.4.3 Texture Evaluation Result

Sample Hair Patches

We performed all the experiments in this section based on the ModiFace hair database. Fig 2.7 shows several patches extracted from the testing images that were used to perform the calculations in this section. As shown in Fig 2.7, some patches are very directional while others are not very directional.

Figure 2.7: Sample hair patches

Main Hair Angle and Standard Deviation of Hair Angles of a Single Hair Patch

In this section, we calculated hair angles and the standard deviation of hair angles in a single hair patch. The hair patch being used in this section is the first hair patch shown in Fig 2.7 and a more zoomed-in version of it is shown in Fig 2.8. The main hair angles and standard deviations calculated using various weighting methods are shown in Table 2.2. An example histogram of the hair angles using gradient magnitude as the weighting function is shown in Fig 2.9.

Figure 2.8: Example hair patch

Table 2.2: Main hair angle and standard deviation of a single hair patch

histogram weighting function	main hair angle	standard deviation
1	79.50	40.23
$ m $	79.50	33.88
$\sqrt{ m }$	79.50	36.98
$\log(1 + m)$	79.50	38.60

Figure 2.9: Histogram using gradient magnitude as bin count

As shown in Fig 2.9, the energy of the histogram peaks at 79 degrees and most of the energy is concentrated inside the interval between 50 and 110 degrees. This information is correctly reflected in the calculated main hair angle and standard deviation values.

When comparing the four weighting functions for standard deviation calculations, while all four weighting functions give the same main hair angle, weighting with gradient magnitude gives the smallest standard

deviation. This makes intuitively sense as this method gives preference to strong edges, and those strong edges give the directionality of hair strands.

Standard Deviation of Hair Angles in Multiple Hair Patches

In this section, we calculated the average of standard deviation of hair angles in hair patches taken from whole images. Before taking the patches, each input image was resized so that the face width in each image was 100 pixels and only patches where every single pixel inside them was hair were taken in each image. In this case, we experimented with hair patches with patch sizes of 4×4 , 8×8 , 12×12 , 16×16 , 24×24 , and 32×32 pixels. The patches were taken from 100 images with different hair style, hair color, and resolution. The reported standard deviation in this section is the average of the standard deviations across all hair patches in all images with the same patch size. The calculated average standard deviations for the different patches are shown in Fig 2.10.

Figure 2.10: Average standard deviation for the different patch sizes

As shown in Fig 2.10, there is a significant increase in the average standard deviation when we increase the patch size from 4×4 to 8×8 pixels for all four weighting functions, then the average standard deviation does not change by much as we continue increasing the patch size. The 4×4 pixels patches have relatively small standard deviations due to the fact that only 4 hair angles are used to calculate the standard deviation. Those four numbers are calculated based on a very small neighbourhood, thus,

tend to agree with each other. As we increase the patch size to 8×8 pixels, 36 hair angles are used to calculate the standard deviation. The local neighbourhood is much larger comparing to that of the 4×4 case, causing it to have a larger standard deviation.

On the other hand, as we keep increasing the patch size, more and more directional structure is revealed. Thus, even though more hair angles are used to calculate the standard deviation as we increase the patch size, more hair angles follow the same pattern (more repetitive hair directions) in a given patch. The contribution of these two factors cancel each other out, causing the standard deviation to remain relatively constant after the 8×8 pixel patch size.

Additionally, similar to the previous section, weighting by gradient magnitude gives the smallest standard deviation in all patch sizes. Its advantage over the non-weighted/weighting by 1 method ("1" method in Fig 2.10) is more significant as the patch size increases. This is due to the fact that there are more directional structure and strong edges in a larger patch (those edges makes hair strand to appear very directional). Therefore, adding more weights to those strong edges results in a smaller standard deviation.

Average Main Hair Angles Calculations

In this section, we calculated the average main hair angles of hair patches in 100 images. We first resized each input image so that the face width in each image was 100 pixels. Then we extracted overlapping patches of either 4×4 or 8×8 pixels, and used the 3×3 Sobel operator to calculate the main hair angle for all hair patches in a single image [35]. Since weighting by gradient magnitude always gave the smallest standard deviation, we only considered this method in the main hair angle calculation in this section. After calculating the main hair angles in all images, we aligned the images based on the face box, overlapped the result we got from the 100 images and calculated the arithmetic mean and standard deviation of places where there were at least 20 main hair angles (those were the places where hair appeared very frequently in the testing images). The standard deviation of the main hair angles at each location represents the confidence we have for main hair angle calculation at that particular location: the lower standard deviation, the higher confidence we have in the main hair angle calculation.

In order to better visualize the results, we constructed a color map for the average main hair angles and standard deviation of the main hair angles. The color map for the average main hair angle is shown in Fig 2.11 and the color map for the standard deviation of the main hair angle is shown in Fig 2.12.

As shown in Fig 2.11, the general trend in the average main hair angle is similar for both patch sizes, except the color regions in the 8×8 pixels color map are less noisy. There is a distinct transition near the central region above the forehead where the main hair angles on the left of the line are less than 90 degrees and the main hair angles on the right of this line are greater than 90 degrees. The exceptions mainly take place around the lower face as hair curls around that face region.

When comparing the standard deviation map in Fig 2.12, the standard deviation of the main hair angle is less on the sides of the face for the 8×8 pixels color map than that of the 4×4 pixels color map. This is mainly due to the fact that when using the 8×8 pixels patches, we are taken more neighbouring information into account, causing the calculation to be more consistent across different images.

Figure 2.11: Top-left: Average main hair angle color map for 4×4 patch size; Top-right: Average main hair angle color map for 8×8 patch size; Bottom: Color Legend, median angle is the median of the directions that a color is responsible for

On the other hand, the standard deviations tend to be bigger near the center top and the bottom edges in both patch sizes. The reason for this is that hair around the two sides of the face are mostly straight, so the main hair angle in most hair styles are around 90 degrees. Hair above the forehead can either go horizontally, horizontally with angles, or pointing downwards (in the case the person has bangs). Furthermore, hair above the forehead usually has a clear transition from hair angle of greater than 90 degrees to hair angles less than 90 degrees, but the place where this transition takes place is highly dependent on the individual hair style. Thus, the uncertainty increases as we go closer to the center of the region above the forehead. The standard deviation is also relatively high near the bottom of the image since as we go further away from the face region, the hair can curl freely in any direction.

2.4.4 Discussion

This section presents the textural patterns in the human hair in terms of direction and distribution of hair flow. We find that the standard deviation of the hair angles increase as we increase the patch size,

Figure 2.12: Top-left: Standard deviation color map for 4×4 patch size; Top-right: Standard deviation color map for 8×8 patch size; Bottom: Color Legend, median angle is the median of the angle range that a color is responsible for

especially from 4×4 to 8×8 pixels patches. We also find there is a clear transition of average main hair angles in the hair region above the forehead, where hair on the left of this line is generally less than 90 degrees and hair on the right of this line is usually greater than 90 degrees. The exception of this rule takes place near the bottom of the face where hair curls around the face. Lastly, we discover that the standard deviation of the main hair angle tend to be lower on the two sides of the face region where there are straight hairs in most hair styles.

2.5 Discussion

In the first part of this chapter, we looked at prior work on neural networks and hair segmentation, with a focus on the ModiFace heuristic-based method [8]. The ModiFace [8] hair segmentation is mainly based on hair color and gradient information of hair samples taken from the input image. This method does not perform equally well for different images and many threshold values have to be determined

experimentally. However, this method sets a good baseline for the other methods described in later chapters.

In the second part of this chapter, we explored texture information in hair patches. Since hair texture follows a certain pattern, we are confident that a well-trained neural network should be able to pick those pattern out. Along with color information, the neural network should be able to classify hair with high accuracy.

Chapter 3

Deep Neural Network (DNN) Based Hair Segmentation

3.1 Proposed Algorithm

Machine learning algorithms have been widely used in several different image segmentation problems [51] [52] [53]. The reason for that is in many cases, even though the object of interest in each segmentation problem stands out from the rest of the image, each input can have different background, lighting, context, etc. Therefore, many algorithms that use color and texture information directly from the input image may work well in certain cases, but perform poorly in other cases.

Supervised classification algorithms usually contain a two stage structure [54]. In the training stage, the algorithm learns how to segment images by looking at different examples of correctly segmented images. In the testing stage, the algorithm applies the rules it learned in the first stage to segment each input image. The rules and features learned in the first stage are automatically generated from the training images instead of hard-coded. Therefore, if the training data set is large enough and is a true representation of the test data, a well-trained model based on machine learning algorithms usually perform well on all new test cases [55].

3.1.1 Model Components

In this chapter, we will discuss two slightly different systems based on a generative model and a discriminative model. The generative model is used as either a feature extractor or a pre-conditioner for the weights in the discriminative model. Before we describe how we combined these two models in the above two scenarios in detail, we will describe the detailed structure of the generative model and discriminative model first.

Generative Model

Generative model belongs to the family of unsupervised learning algorithms. The idea of this model is that after seeing many examples of the hair and non-hair patches, the network can figure out a good interpretation for each kind of input. Therefore, the goal of the training stage is to tune the weights in the network to find a good representation of the input.

There are many different generative models, such as the Gaussian mixture model and the hidden Markov model. In this section, we use a generative model called the Restricted Boltzmann machine (RBM). RBM is more suitable in this case since it is very easy to train and it can be used to pre-train the neural networks. A Boltzmann machine is an energy based model that is constructed by connecting binary stochastic neurons using symmetric connections. A Restricted Boltzmann machine is the simplest Boltzmann machine where it only has one layer of hidden units, the hidden units are conditionally independent given the visible states, and there are no connections between units in the same layer. The assumption of independence between hidden units simplifies the training procedure and allows training to be done more quickly.

The training samples to the RBM are generated in the following way: For each input image in the training set, we generate image patches of a fixed size (e.g. 4 x 4 pixels) using a sliding window. For each image patch, we do the following:

1. Normalize the pixel values of R, G, B channel by dividing the values by 255.
2. Concatenate the normalized pixel values in each of the R, G, B channel to construct three vectors, one for each channel.
3. Concatenate the three vectors constructed in Step 2 into a single vector. This vector is used as an input vector to the RBM.
4. Based on the corresponding pre-cut hair mask for the input image, label the current image patch as a hair patch or non-hair patch.

The RBM is trained with Contrastive Divergence gradient estimator with 1 full Gibbs update (CD-1) [56] by performing the steps below:

1. Clamp an input vector on the visible units and set the hidden units to random binary states.
2. Sample a binary state for the hidden units conditional on the visible units (v_i, h_j^0) . Since the input in this case is not binary, but a real value between 0 and 1, each value in the input vector represents the probability that a particular visible unit is turned on.
3. Sample a binary state for the visible units conditional on the binary hidden state (reconstructed visible state).
4. Sample a binary state for the hidden units conditional on the reconstructed visible state (v_i, h_j^1) .

we repeat this procedure for all data vectors in the mini-batch and find the expected value of v_i, h_j^0 ($\langle v_i, h_j^0 \rangle$) and v_i, h_j^1 ($\langle v_i, h_j^1 \rangle$). Then the weight connecting a visible unit i and a hidden unit j is

updated using the equation [56]:

$$\Delta w_{ij} = \epsilon(<v_i, h_j>^0 - <v_i, h_j>^1) \quad (3.1)$$

After the training stage, the weights of the RBM are stored. The weights are to be used later on by the discriminative model.

Discriminative Model

The goal of the discriminative model is to classify the inputs as hair or non-hair (the model now only handles binary classification since the current hair database has slight mis-alignment between the image and the pre-cut mask. This mis-alignment has minor effect when determining the performance of different models, but it makes it difficult to generate partially hair patches from the training images). The general structure of a neural network is shown in Fig 3.1.

During the training stage, the training samples are first shuffled into random order and then feed into the neural network one by one. Other than the output layer, a bias unit is added to each layer of the network. The input for each hidden unit and output unit is generated by summing together all the inputs and bias connected to that unit and then passing into an activation function. The activation function to all the hidden units is the rectified linear unit (ReLU) activation function and the activation function to the output units is the SoftMax function. The neural network is trained with the usual back-propagation algorithm with a mini-batch size of 100. The learning rate is initialized to a relatively larger number and decreases in each epoch to prevent random oscillations as the optimization process gets closer to a local optimum. At the same time, momentum is also used in the training procedure (increase the step size if we travel in the same direction as the previous direction).

In order to prevent over-fitting to the training samples, several regularization techniques are used to train the neural networks. First of all, early stopping (tracking the error in the validation set and stop the training process if the validation error starts to increase) is applied to prevent over-training of the neural networks. Additionally, dropout technique [57] is also applied to the neural network. A 50% dropout rate is applied to each hidden layer of the neural network (randomly turn off half of the hidden units in a hidden layer for each training case). Dropout technique can effectively avoid over-fitting since it is equivalent of training multiple models with different units turned on at the same time and the output is the average of the predictions produced by those models. However, the training complexity does not increase in this case since all the models share the same set of weights.

3.1.2 Using Generative Model as a Feature Extractor

An overview of the proposed algorithm is shown in Fig 3.2. For the hair segmentation task, it is difficult to define the features that best separate hair from the rest of the image (since hair can have any color and even the same hair sample can show up differently in terms of color and texture under different lighting conditions and scales). Therefore, our proposed algorithm first uses a generative model to find features from the input images, and then passes those features into a neural network for further classification.

Figure 3.1: Multilayer neural network model architecture

In other words, we first train an RBM with the training set and save the weights of the trained RBM. When we train the neural network, we first pass the raw inputs into the trained RBM, and the outputs of RBM are passed into a neural network as inputs.

3.1.3 Using Generative Model to Pre-train the Weights of the Neural Network

It has been found in the past that it is very difficult to train a deep neural network as its cost function is highly non-convex [19]. This non-convexity property makes it very easy to get stuck at local minima that do not perform well on the testing set [19]. However, if we pre-train the weights of the neural network with a generative model instead of initializing them randomly, the deep neural network is able to achieve much better performance [19].

The most common approach to pre-train the neural network is to first train a deep belief net (DBN)

Figure 3.2: System overview: a) Take patches from the input image b) Vectorize the patch c) Extract features with the generative model d) Classify the features with neural networks

and then use the weights of the deep belief net to initialize the weights in the neural network [56]. A deep belief net is constructed by stacking several RBM together and it can be trained with a greedy approach [56]: we train one layer of deep belief net at a time from the bottom up (starting from the input layer) using the same procedure as training an RBM. When we train the weights at one layer, we assume all the other weights are fixed. We repeat this procedure for all the layers in the deep belief net. The number of layers in the deep belief net depends on the depth of the neural network, since the weights of the deep belief net are used to initialize the weights in the neural network.

Then once we have initialized the weights in the neural network, we can just train the neural network using the back-propagation algorithm (using the same inputs as in the RBM training).

3.1.4 Post-processing

During the testing stage, one effective way to reduce false positives is to apply a hair occurrence prior probability model to the output of the neural network. An example of the hair occurrence prior probability model is shown in Fig 2.5. The prior model is produced by first re-sizing the face bounding box of each pre-cut hair mask to the following dimensions: left-x = 200th pixel, top-y = 200th pixel, right-x = 300th pixel, bottom-y = 333th pixel. Then the re-sized pre-cut masks are super-composed together and normalized to represent the probability of a certain location being hair. A threshold value of 0.1 is used in this case to turn the prior model into a binary prior mask. We used a total of 115 hair masks taken from the ModiFace database to produce the prior hair mask. When applying the prior mask, it is first re-sized to match the size of the face bounding box in the test image, and then combined with the neural network result to produce the final output.

3.2 Experimental Result

3.2.1 Dataset

All the training and testing images were taken from the ModiFace database consisting of 115 images along with the corresponding cut-out hair mask for each image (produced by manual hair segmentation). There are both high-quality images (professionally-shot images) and low-quality images (user-uploaded images) in the database. The photos in the database were all taken under different backgrounds, light conditions, and resolutions. Four sample images and their corresponding hair cut-outs are shown in Fig 3.3.

For each experiment performed in this chapter, the 115 images are randomly divided into two sets: a training set (58 images) and a test set (57 images). The patches taken in the training set are further divided as training set (80%) and validation set (20%). The training set is used to train the models, and the validation set is used to select the best hyper-parameters for a model. The training patches are taken from each training image with an overlapping sliding window. The test images are divided into non-overlapping patches with the same size as the training patches. In order to make the experiments more robust to outliers, the above procedure is repeated 10 times for each experimental set-up and the average performance is reported as the final performance measure.

3.2.2 Facial Coordinates Detection

Before applying the prior mask, we need to match the face bounding box of the input image and that of the prior probability mask. We used the algorithm described in [33] to detect facial coordinates.

3.2.3 Performance Measure

We measure the performance of the models based on two quantities: performance score and F1 score.

Figure 3.3: Top row: Sample images from the database; Bottom row: Corresponding hair cut-out

The performance score is calculated based on two quantities: accuracy and over-estimation. Accuracy measures the percentage of the pre-cut hair mask (ground truth) that was captured by the output produced by the proposed algorithm. Over-estimation measures the percentage of pixels that were wrongly classified as hair pixels by the proposed algorithm comparing to the pre-cut hair mask. To better illustrate the idea of accuracy and over-estimation, we calculate the accuracy and over-estimation with a graphical example shown in Fig 3.4. Assume the positive ground truth we are trying to detect is a blue square (we use symbol t to represent it), the correctly detected positive region is in purple (we use symbol d to represent it), and the wrongly classified positive samples (they are negative regions in the ground truth) are in red (we use symbol e to represent it), then accuracy = d/t and over-estimation = e/t .

The best model has the highest score based on the following equation:

$$p = (a - e) \times a \quad (3.2)$$

where p is the performance score, a is accuracy, and e is over-estimation.

The performance score is a good measure of the performance since we want a high accuracy and a low over-estimation. However, in many cases, the difference between the accuracy and over-estimation is the same for many different model settings. In this case, we favour high accuracy over low over-estimation since the goal of the system is to detect hair in the input images.

Figure 3.4: Graphical illustration of accuracy and over-estimation: accuracy = d/t and over-estimation = e/t

The F1 score is a standard performance measure based on precision and recall [58]. The precision and recall can be calculated based on the confusion matrix shown in Table 3.1 [58]:

$$pre = \frac{TP}{TP + FP} \quad (3.3)$$

$$rec = \frac{TP}{TP + FN} \quad (3.4)$$

where *pre* is precision and *rec* is recall.

In other words, precision represents the fraction of detected positive response are correct and recall represents the fraction of ground truth being detected.

Table 3.1: Confusion Matrix

	Classified +	Classified -
True +	TP	FN
True -	FP	TN

After getting the precision and recall score, the F1 score is calculated as follows:

$$F1 = 2 \times \frac{pre \times rec}{pre + rec} \quad (3.5)$$

3.2.4 Experiments for Using Generative Model as a Feature Extractor

Since pre-training with a generative model takes longer time to run, we first use the generative model as a feature extractor for the neural network to find an optimal set-up for variables such as sample ratio and patch size.

Effect of the prior probability model

In this section, we first want to demonstrate the effectiveness of applying the prior probability model in the post-processing step. We experimented with a positive (hair sample) to negative (non-hair sample) ratio of 1:1, 1:2, 1:3, and 1:4 in the training samples. In each case, the total number of positive samples is fixed as 20,000 patches. We also experimented with three different patch sizes: 4×4 pixels, 8×8 pixels and 16×16 pixels. Since the goal of this section is to demonstrate the effectiveness of the prior probability model, neural networks are not used at this stage. Instead, after getting the features from the RBM, they are summed and passed into a sigmoid function to produce the final classification output. The detailed performance is shown in Table 3.2 and the performance score comparison is shown in Table 3.3.

Table 3.2: Accuracy and over-estimation comparison on the effect of the prior probability model

		patch size					
		4×4 pixels		8×8 pixels		16×16 pixels	
+ to - sample ratio		accuracy	over-est	accuracy	over-est	accuracy	over-est
1 to 1	no prior	0.7783	1.6254	0.8112	1.8064	0.7691	1.5921
	with prior	0.7207	0.7589	0.7515	0.8516	0.7113	0.7557
1 to 2	no prior	0.6260	0.9249	0.6244	0.9185	0.5993	0.7191
	with prior	0.5784	0.4352	0.5786	0.4368	0.5531	0.3533
1 to 3	no prior	0.3909	0.5410	0.4830	0.6173	0.4288	0.4600
	with prior	0.3578	0.2372	0.4445	0.2832	0.3957	0.2276
1 to 4	no prior	0.2380	0.2290	0.3216	0.2982	0.3885	0.3571
	with prior	0.2151	0.1083	0.2949	0.1417	0.3594	0.1743

Table 3.3: Performance score comparison on the effect of the prior probability model

		patch size		
positive to negative sample ratio		4×4 pixels	8×8 pixels	16×16 pixels
1 to 1	no prior	-0.6593	-0.8073	-0.6330
	with prior	-0.0275	-0.0752	-0.0315
1 to 2	no prior	-0.1871	-0.1837	-0.0718
	with prior	0.0828	0.0821	0.1105
1 to 3	no prior	-0.0587	-0.0649	-0.0134
	with prior	0.0431	0.0717	0.0665
1 to 4	no prior	0.0021	0.0075	0.0122
	with prior	0.0230	0.0452	0.0665

As shown in Table 3.2, the over-estimation is almost always greater than the accuracy when we do not apply the prior probability model (except for sample ratio of 1 to 4). After we apply the prior probability mask, the accuracy does not decrease by much, but the over-estimation is reduced by around 50%, causing the over-estimation to be smaller than accuracy in most cases. Furthermore, as shown in Table 3.3, the performance score is always greater after we apply the prior probability model in the post-processing step. Thus, the prior probability model is effective in reducing errors in the detected hair mask and we apply this post-processing step in all experiments after this section.

Finding the Optimal Sample Ratio and Patch Size

In this section, we experimented with a positive (hair sample) to negative (non-hair sample) ratio of 1:1, 1:2, 1:3, and 1:4 in the training samples. In each case, the total number of positive samples is fixed as 20,000 patches. We also experimented with three different patch sizes: 4×4 pixels, 8×8 pixels and 16×16 pixels. Since the best sample ratio and patch size has the most significant effect on the features generated by the generative model, neural networks are not used at this stage. Instead, after getting the features from the RBM, they are summed and passed into a sigmoid function to produce the final classification output. The accuracy and over-estimation on the test images (after applying the prior mask) are shown in in Table 3.4 and the precision and recall results can is shown in Table 3.5.

Table 3.4: Accuracy and over-estimation for different sample ratios and patch sizes

patch size		positive to negative sample ratio			
		1 to 1	1 to 2	1 to 3	1 to 4
4×4 pixels	accuracy	0.7207	0.5784	0.3578	0.2151
	over-estimation	0.7589	0.4352	0.2372	0.1083
8×8 pixels	accuracy	0.7515	0.5786	0.4445	0.2949
	over-estimation	0.8516	0.4368	0.2832	0.1417
16×16 pixels	accuracy	0.7113	0.5531	0.3957	0.3594
	over-estimation	0.7557	0.3533	0.2276	0.1743

Table 3.5: Precision and recall for different sample ratios and patch sizes

patch size		positive to negative sample ratio			
		1 to 1	1 to 2	1 to 3	1 to 4
4×4 pixels	precision	0.4871	0.5707	0.6013	0.6651
	recall	0.7207	0.5784	0.3578	0.2151
8×8 pixels	precision	0.4688	0.5698	0.6108	0.6755
	recall	0.7515	0.5786	0.4445	0.2949
16×16 pixels	precision	0.4849	0.6102	0.6348	0.6733
	recall	0.7113	0.5531	0.3957	0.3594

As shown in Table 3.4, as the percentage of the negative samples increases in the training samples, both the accuracy and over-estimation decreases. The reason for this is as the number of negative samples increases, the model is able to achieve a lower error rate if it favours negative samples (since the probability of a negative sample appearing is higher than that of a positive sample). This suggests that we should include equal number of positive and negative training samples. However, in a natural image, there are more negative samples than positive samples if we use a sliding window to take all the possible patches from an image (moving the sliding window by 1 pixel each time). Furthermore, there are many more variations among negative samples than in positive samples, so the model should ideally see more negative samples than positive samples in order to capture good representations of the negative samples. Thus, it is crucial to find an optimal ratio between the positive and negative samples for the training stage that take into account the restrictions mentioned above.

At the same time, as shown in Table 3.5, as the percentage of the negative samples increases in the training samples, precision increases but recall decreases. This is due to the fact that as we increase the number of negative training samples, the neural network becomes less aggressive in detecting hair and the neural network is more certain about the positive samples it detects. This increases the precision

(the fraction of the positive response that is correct) and decreases the recall (the fraction of ground truth being detected).

In the meanwhile, once the sample ratio is fixed, we also experimented with different patch sizes. As shown in Table 3.4, the accuracy and over-estimation also varies among different patch sizes for the same sample ratio. When the patch size is too small, it might not contain enough texture/color information for the model to decide if the current patch is hair or not. On the other hand, if the patch size is too big, many patches in the test images will become partial hair patches (those patches contain both hair and background pixels). Since the training samples do not cover such cases, the trained model will likely to classify those samples wrong, reducing the detection accuracy.

When we fixed the sample ratio, the precision and recall do not change by too much for different patch sizes. This shows that precision and recall are not affected too much by patch size.

It is very difficult to find the best setting for the sample ratio and patch size based on the above analysis. We need to calculate the performance score and F1 score for each set-up to determine the optimal model. The performance score for sample ratio 1:1, 1:2, 1:3 and 1:4 is shown in Table 3.6 and the F1 score is shown in Table 3.7. The optimal setting for the performance score is to use a sample ratio of 1:2 and a patch size of 16×16 pixels and its performance score is much higher than any other settings. The optimal score for the F1 score is to use a sample ratio of 1:1 and a patch size of 4×4 pixels. This setting results in a very aggressive neural network (causing a very high recall and a very low precision). At the same time, the F1 score at a sample ratio of 1:2 and a patch size of 16×16 pixels gets the second highest F1 score (very close to the highest F1 score). Taking both the performance score and F1 score into account, the most optimal setting overall is a sample ratio of 1:2 and a patch size of 16×16 pixels.

Table 3.6: Performance score for models with different sample ratios and patch sizes

	positive to negative sample ratio			
patch size	1 to 1	1 to 2	1 to 3	1 to 4
4×4 pixels	-0.0275	0.0828	0.0431	0.0230
8×8 pixels	-0.0752	0.0821	0.0717	0.0452
16×16 pixels	-0.0315	0.1105	0.0665	0.0665

Table 3.7: F1 score for models with different sample ratios and patch sizes

	positive to negative sample ratio			
patch size	1 to 1	1 to 2	1 to 3	1 to 4
4×4 pixels	0.5813	0.5745	0.4486	0.3250
8×8 pixels	0.5774	0.5742	0.5145	0.4106
16×16 pixels	0.5767	0.5803	0.4875	0.4686

Finding the Optimal Parameters for the Discriminative Model

After finding the optimal sample ratio and patch size, the next step is to find the optimal number of hidden layers in the neural networks. The performance of the neural network does not necessarily improve as more hidden layers are included in the model since more hidden layers might cause over-fitting to the

training data and also more hidden layers require more training effort. In this section, we experimented with models with up to 4 hidden layers, each layer has 1000 hidden units. The performance score for each setting is shown in Table 3.8 and the F1 score is shown in Table 3.9. Zero hidden layers refers to the model that sums the output from the RBM directly to get the classification result.

As shown in Table 3.8 and Table 3.9, the model with 4 hidden layers achieves the best score. The reason for this is with the same number of units in each hidden layer, a model with more hidden layers has larger capability assuming over-fitting has not occurred. In this case, since dropout technique is applied to each hidden layer, over-fitting has not occurred so far. On the other hand, the performance improvement from the model with 3 hidden layers and 4 hidden layers is extremely small. This means that the payoff (increase in performance) for the increasing complexity of the model diminishes as we keep increasing the number of hidden layers.

Table 3.8: Performance score for different number of hidden layers, each layer has 1000 hidden units

number of hidden layers	accuracy	over-estimation	performance score
0	0.5531	0.3533	0.1105
1	0.5927	0.3954	0.1169
2	0.5833	0.3975	0.1084
3	0.6226	0.4251	0.1230
4	0.6304	0.4312	0.1256

Table 3.9: F1 score for different number of hidden layers, each layer has 1000 hidden units

number of hidden layers	precision	recall	F1 score
0	0.6102	0.5531	0.5803
1	0.5998	0.5927	0.5962
2	0.5948	0.5833	0.5890
3	0.5943	0.6226	0.6081
4	0.5938	0.6304	0.6116

We also experimented with different number of hidden units per layer for a neural network with 4 hidden layers and the result is shown in Table 3.10 and Table 3.11 . Initially, the performance of the system increased significantly when the number of hidden units is increased from 500 to 1000. Then the system has a similar performance when the number of hidden units is further increased from 1000 to 1500. Similar to the experiment with different number of hidden layers, the initial payoff for increasing the number of units is higher since we increase the capacity of the model. However, the payoff diminishes as we keep increasing the complexity of the model since the model already has enough capacity to handle the variations in the inputs.

Table 3.10: Performance score for different number of hidden units per layer in a four-hidden-layers network

number of hidden units	accuracy	over-estimation	performance score
500	0.5966	0.4260	0.1018
1000	0.6304	0.4312	0.1256
1500	0.6260	0.4303	0.1225

Table 3.11: F1 score for different number of hidden units per layer in a four-hidden-layers network

number of hidden units	precision	recall	F1 score
500	0.5834	0.5966	0.5900
1000	0.5938	0.6304	0.6116
1500	0.5926	0.6260	0.6089

Effect of Feature Extraction with Generative Model

The performance comparison for a neural network with 1 hidden layer and 1000 hidden units with and without the generative model (raw pixel is fed into the neural network directly without the generative model feature extractor) is shown in Table 3.12. With the generative model feature extraction, the performance score increases from 0.0724 to 0.1169 and the F1 score increases from 0.5603 to 0.5962 (with a sample ratio of 1:2 and a patch size of 16×16 pixels). Therefore, the feature extraction step is effective in increasing the performance of the system.

Table 3.12: Performance comparison on the effect of the generative model for a neural network with 1 hidden layer and 1000 hidden units

	with generative model	without generative model
accuracy	0.5927	0.5539
error	0.3954	0.4232
performance score	0.1169	0.0724
precision	0.5998	0.5669
recall	0.5927	0.5539
F1 score	0.5962	0.5603

Run time comparison

We performed all the experiments on a 2.7 GHz Intel i5 CPU. The training time mentioned in this section contains three components: getting patches from the training images and reshaping them into vectors, training the generative model, and training the neural network. The positive to negative sample ratio used in this section was 1:2 (i.e. 60,000 patches were extracted from the training images in total), and patches extracted had dimensions of 16×16 pixels. During the training process, the generative model went through the training samples 50 times (50 epochs) and the neural network went through the training samples at most 150 times (150 epochs). In case of early stopping, the neural network still needs to go through at least 50 epochs. The training time for different number of hidden layers is shown in Table 3.13 and Fig 3.5.

Since the same output from the generative model was used as the input to the neural network, the time spent on input generation and training the generative model was fixed across different neural network architectures. When the number of hidden units is fixed, the training time scales linearly with the number of hidden layers. This is mainly due to the fact that the time spent to train the neural networks scales linearly with the number of hidden layers and training the neural network takes significantly longer than generating inputs and training the generative model (which have constant time in this case).

The training time for different number of hidden units is shown in Table 3.14 and Fig 3.6. When the

Table 3.13: Training time for neural network with different number of hidden layers, each layer has 1000 hidden units

	number of hidden layers			
	1	2	3	4
generating training inputs (hr)	0.42	0.42	0.42	0.42
training generative model (hr)	0.62	0.62	0.62	0.62
training neural network (hr)	2.94	5.41	7.88	9.77
total (hr)	3.98	6.44	8.92	10.81

Figure 3.5: Training time for neural network with different number of hidden layers, each layer has 1000 hidden units

number of hidden layer is fixed, the training time scales linearly with the number of hidden units.

Table 3.14: Training time for neural network with different number of hidden units, with 4 hidden layers

	number of hidden units				
	500	750	1000	1250	1500
generating training inputs (hr)	0.42	0.42	0.42	0.42	0.42
training generative model (hr)	0.62	0.62	0.62	0.62	0.62
training neural network (hr)	3.02	6.38	9.77	13.50	19.62
total (hr)	4.06	7.42	10.81	14.54	20.66

On the other hand, the testing time has 5 different components: splitting the testing image into non-overlapping patches and reshaping them into vectors, applying the trained generative model, applying the trained neural network, constructing the output image based on the outputs from the neural network, and applying the post-processing step. The testing time for a single input image for different number of hidden layers in the neural network is shown in Table 3.15 and for different number of hidden units is shown in Table 3.16. The testing time remains relatively constant in all different setting of the neural network. The reason for this is image splitting and combining the outputs from the neural network play the most dominant role in the total testing time, and the time spent on those two stages are similar

Figure 3.6: Training time for neural network with different number of hidden units, with 4 hidden layers

across all model settings.

Table 3.15: Testing time (per image) for different number of hidden layers in the neural network, each layer has 1000 hidden units

	number of hidden layers			
	1	2	3	4
image splitting (sec)	0.83	0.83	0.83	0.83
applying generative model (sec)	0.27	0.27	0.27	0.27
applying neural network (sec)	0.02	0.03	0.04	0.05
constructing output image (sec)	0.86	0.89	0.92	0.94
applying prior mask (sec)	0.56	0.55	0.63	0.60
total (sec)	2.55	2.57	2.69	2.69

Table 3.16: Testing time (per image) for different number of hidden units in the neural network, with 4 hidden layers

	number of hidden units				
	500	750	1000	1250	1500
image splitting (sec)	0.83	0.83	0.83	0.83	0.83
applying generative model (sec)	0.27	0.27	0.27	0.27	0.27
applying neural network (sec)	0.02	0.04	0.05	0.07	0.09
constructing output image (sec)	0.86	0.86	0.94	0.86	0.91
applying prior mask (sec)	0.58	0.53	0.60	0.56	0.57
total (sec)	2.56	2.53	2.69	2.59	2.67

3.2.5 Experiments for Using Generative Model to Pre-train the Neural Network Weights

In this section, we compare the performance of a deep network with using RBM as a feature extractor and a deep neural network with pre-trained weights using a deep belief net (with 3 and 4 hidden layers, with 1000 hidden units in each layer). The detailed performance comparison is shown in Table 3.17.

Table 3.17: Performance comparison with and without neural network weights pre-training

	3 hidden layers		4 hidden layers	
	feature extraction	weights pre-training	feature extraction	weights pre-training
accuracy	0.6226	0.6500	0.6304	0.6429
error	0.4251	0.4399	0.4312	0.4100
performance	0.1230	0.1366	0.1256	0.1497
precision	0.5943	0.5964	0.5938	0.6106
recall	0.6226	0.6500	0.6304	0.6429
F1	0.6081	0.6220	0.6116	0.6263

As shown in Table 3.17, the performance score increases from 0.1230 to 0.1366 for a neural network with 3 hidden layers and from 0.1256 to 0.1497 for a neural network with 4 hidden layers after we pre-train the weights with a deep belief net. Therefore, pre-training the weights with a deep belief nets is effective in increasing neural network performance.

Run-time Comparison

We performed all the experiments on a 2.7 GHz Intel i5 CPU. The training time mentioned in this section contains three components: getting patches from the training images and reshaping them into vectors, training the generative model, and training the neural network. The positive to negative sample ratio used in this section was 1:2 (i.e. 60,000 patches were extracted from the training images in total), and patches extracted had dimensions of 16×16 pixels. During the training process, the generative model went through the training samples 50 times (50 epochs) and the neural network went through the training samples 150 times (150 epochs). In case of early stopping, the neural network still needs to go through at least 50 epochs. The training time comparison for using RBM as a feature extractor and using deep belief net to pre-training the weights is shown in Table 3.18 (the neural network has 4 hidden layers, each layer consists of 1000 hidden units).

Table 3.18: Training time comparison for pre-training a neural network with 4 hidden layers, each layer consists of 1000 hidden units

	with feature extraction	with weights pre-training
generating training inputs (hours)	0.42	0.42
training generative model (hours)	0.62	2.48
training neural network (hours)	9.77	9.77
total (hours)	10.81	12.67

The only difference in the training time is caused by the training the generative model since instead of training a RBM to be used as a feature extractor, we need to train a 4 layers deep belief net (which takes 4 times longer) to be used as the initial weights in the neural network. Using a deep belief net

to pre-train the weights takes 17% more in the total training time comparing to using the RBM as a feature extractor.

However, the testing time remains the same for both cases.

3.2.6 Sample Output

Four successfully segmented hair samples are shown in Fig 3.7 and four failure cases are shown in Fig 3.8.

When comparing the classification results for professionally-shot images with user-uploaded images, the performance for professionally-shot photos is much better in general. This is mainly due to the fact that user-uploaded photos have much larger variations in terms of background, lighting condition, and viewing angle. Therefore, the hair segmentation with user-uploaded photos is much more difficult in comparison. Using more user-uploaded photos in the training stage might help to improve the performance since it can help to capture many more hair variations in the user-uploaded photos.

Figure 3.7: Successfully segmented hair samples. Top row: input image; Bottom row: output from the proposed algorithm (shown as red pixels)

Figure 3.8: Sample failure cases. Top row: input image; Bottom row: output from the proposed algorithm (shown as red pixels)

3.3 Discussion

In this section, we experimented with hair segmentation with neural networks. We experimented with neural networks with different architecture as well as different pre-training method. We mainly discussed two slightly different systems based on a generative model and a discriminative model. The first system first uses an RBM to generate features from patches taken from the input image and then passes those features into a neural network for classification. The second system first trains the raw patches with a deep belief net and then uses the weights in the deep belief net to initialize the weights of the neural network. The outputs from the neural network are combined with a hair occurrence prior probability model to produce the final output. Both models are tested with a database that contains images with different hairstyle, hair color, resolution, and background.

Both systems explored in this section have very high training time and very low testing time. However, the overall performance of the neural network systems discussed in this chapter is not very good. In order to improve the performance of the neural network, many more training images need to be used. However, more training images are not easy to obtain since the most accurate way to produce the ground truth hair mask for an image is manual segmentation. Therefore, producing many more training images is a very tedious and time-consuming task.

Chapter 4

Heuristically-Trained Neural Network (HNN) Hair Segmentation

As seen in the previous chapter, training a universal neural network for hair segmentation did not perform very well in our case since this method requires a large number of training images. Instead, in this chapter, we train a neural network for each input image based on the information we collected from a heuristic-based method.

The intuition behind the heuristically-trained neural network is to utilize human-defined criteria when taking training samples for the neural network. A human being can easily distinguish between hair and non-hair regions in an image based on mainly color and texture information. Similarly, based on human-defined criteria on color and texture, the heuristic-based method can already identify certain hair and non-hair regions with high confidence. We then use those high confidence regions to train a neural network, aiming to apply what it learns from this set to classify the rest of the image, especially the regions where human-defined criteria cannot effectively classify. In other words, based on the human-defined criteria, a neural network is trained and is then used to classify areas which the human-defined criteria could not.

The idea of selecting the most representative training data to achieve better training efficiency is widely used in active learning [59] [60] [61]. Specifically in [62], the author applies clustering to the training samples so that the classifier only needs to classify the representative samples in each cluster (since samples belong to the same cluster are likely to have the same label). The author proposes that the representative samples are usually either the ones in the cluster centers and/or the ones on the boundary of the clusters. Our algorithm is similar to [62] in a sense that it first divides the input images into patches that we can classify with high confidence and the ones that we cannot classify effectively. Then we train a neural network based on the patches in the high confidence cluster (representative samples), aiming to classify the patches that we could not effectively classify before. An overview of HNN steps is shown in Fig 4.1 to Fig 4.5.

Figure 4.1: Initially: Data is not labelled or arrange in any way

Figure 4.2: Step 1: We apply a heuristic rule to sort data

4.1 Training Set Generation

Since we aim to train a neural network for each image, the positive and negative training samples are taken from each input image based on the heuristic-based method mentioned in section 2.2 (We call

Figure 4.3: Step 2: We separate data into high-confidence regions and low-confidence regions

Figure 4.4: Step 3: We use high-confidence positive and negative samples to train a neural network

it the "ModiFace [8]" method in this chapter). To get the positive training samples, we use a larger threshold so that we are more certain that the samples we get are all hair patches. On the other hand,

Figure 4.5: Step 4: We use the trained neural network to classify all data

we use a smaller threshold for the negative training samples since we do not want to include any hair patches in the negative samples.

4.1.1 Positive Training Sample Generation

To obtain the positive training samples, we first select a threshold that is greater than or equal to 0.5 and create a hair mask using the ModiFace [8] method (we call it the "big hair mask" and several samples of the big hair mask are shown in Fig 4.6). Then we divide the input image into non-overlapping patches. For each patch, we check against the big hair mask and include it as a positive sample if all pixels in that patch are hair pixels according to the big hair mask. Due to the larger threshold that is used to generate the positive samples, we often end up with a very small set of positive samples. However, since hair usually follows similar pattern but the hair pattern may orient differently depends on the location relative to the face, we can generate more positive samples by rotating the hair patches. Thus, in order to increase the number of positive samples, we can generate more plausible positive samples by rotating the input image and hair mask by n degrees each time, from 0 to 360 degrees, where n is the rotation step (rotation step = 5 by default). We then take the positive patches from the rotated images.

4.1.2 Negative Training Sample Generation

To obtain the negative training samples, we first select a threshold that is less than or equal to 0.5 and create a hair mask using the ModiFace [8] method (we call it the "small hair mask" and several samples

Figure 4.6: Sample hair masks. First row: input image; Second row: big hair mask; Third row: small hair mask; Fourth row: negative samples generated with small hair mask

of the small hair mask are shown in Fig 4.6). Then we divide the input image into non-overlapping patches. For each patch, we check against the small hair mask and include it as a negative sample

if all pixels in that patch are non-hair pixels according to the small hair mask. We do not introduce any rotation to generate more negative samples since one non-hair patch is rarely a rotated version of another non-hair patch.

4.1.3 Training Set Generation

After getting the positive and negative samples, we want to mix them up to form a training set. In order to avoid sampling bias in the neural network, we want to include the same number of positive and negative samples in the training set. However, it is rarely the case that we end up with exactly the same number of positive and negative samples. Therefore, we generate the training set using the following rules:

1. Take turns to draw samples (without replacement) from positive set and negative set so that we always see a positive sample after a negative sample and vice versa.
2. Always draw negative samples randomly from the negative set.
3. When drawing positive samples, always give preference to the non-rotated samples: If there are still non-rotated positive samples in the positive set, then draw one of the non-rotated samples; when there are no non-rotated samples left, then draw a rotated positive sample (randomly).
4. Assume there are n negative training samples and m positive training samples. If we have used up all the samples in the negative set, we reset the negative set to the original negative set (the pool that we select negative samples from has n samples again). If we have used up all the samples in the positive set, we reset the positive set to the original positive set (the pool that we select positive samples from has m samples again). In other words, if $n > m$, then we will use some of the positive samples multiple times in the training set.
5. For minimal training set: Stop drawing samples once we have used up all the positive set or the negative set.
6. For maximal training set: Stop drawing samples once we have used up all the positive set and the negative set.

4.2 Experimental Results

4.2.1 Neural Networks Structure

In this section, we always divided images into 4×4 pixels patches, so that the input layer of the neural network had 48 nodes. We experimented with a neural network with one hidden layer with 96 nodes. The structure of the neural network was the same as the one described in section 3.1.1. The neural network outputted the probability of a patch being hair. We later set a threshold to turn the neural network output into a binary hair mask.

4.2.2 Performance Measure

Since we performed training directly on individual images, we did not need to divide the database into training set and test set in this section. We performed our testing on 100 images taken from the ModiFace database (same database as the one used in the previous chapter, but we selected 100 images out of the 115 images). The testing result consists of two sets of data: the performance score and the F1 score (as described in section 3.2.3).

4.2.3 Hair Mask Constructed Based on Heuristically-Trained Neural Network Output

In this section, we focus on the performance of hair mask constructed based on HNN output. In addition to the HNN output, some post-processing steps (cropping eye regions, finding the largest connected component, gap filling, and hair mask blurring mentioned in Section 2.2) are applied to improve the performance of the final hair mask.

There are three key threshold values when constructing the HNN hair mask:

1. big threshold (α_{thres}): This is the final threshold that is used to threshold the total score of a ModiFace [8] hair mask to turn it into a binary mask. The big threshold results in a conservative hair mask and is used to generate positive training samples for the HNN.
2. small threshold (β_{thres}): This is the final threshold that is used to threshold the total score of a ModiFace [8] hair mask to turn it into a binary mask. The small threshold results in an aggressive hair mask and is used to generate negative training samples for the HNN.
3. best threshold (γ_{thres}): This is the final threshold that is used to threshold the output of the HNN to turn it into a binary mask.

Training with the Ground Truth

In this section, we started by using the hair cut-out to generate training samples, aiming to find the best result achievable using this set-up. In this case, we used the hair cut-out as the big hair mask to generate the positive samples and also used it as the small hair mask to generate the negative samples. The best achievable performance using this set-up at the best threshold (γ_{thres}) is shown in Table 4.1 (with a minimal training set).

Table 4.1: Detailed performance of heuristically-trained neural network hair masks generated based on training with ground truth

	accuracy	over-estimation	performance	precision	recall	F1
HNN	0.8722	0.2381	0.5531	0.7856	0.8722	0.8266
ModiFace [8]	0.7274	0.2281	0.3632	0.7613	0.7274	0.7439

As shown in Table 4.1, the performance of the hair mask based on the HNN output is quite good. Comparing to the ModiFace [8] method mentioned in Section 2.2, the performance score increases from

0.3622 to 0.5531. In other words, the best achievable performance with this set-up is 0.5531.

Some sample HNN hair mask outputs based on training samples taken from the ground truth are shown in Fig 4.7.

Figure 4.7: Sample outputs: First row: input image; Second row: ModiFace [8] hair mask output ; Third row: HNN hair mask output

Finding the Optimal Threshold Values

The performance of the HNN is highly dependent on the training samples that were used to train it. In this section, we experimented with the different combinations of the small thresholds (β_{thres}) and the big thresholds (α_{thres}), with a minimal training set. The performance of different combination of

thresholds for hair mask based on the HNN output is shown in Table 4.2 and Table 4.3.

Table 4.2: Performance score for hair masks based on the HNN output

	big threshold (α_{thres})					
small threshold (β_{thres})	0.5	0.55	0.6	0.65	0.7	0.75
0.1	0.3312	0.3174	0.3378	0.3545	0.3474	0.3122
0.15	0.3188	0.3360	0.3384	0.3363	0.3522	0.3441
0.2	0.3357	0.3364	0.3433	0.3497	0.3554	0.3734
0.25	0.3437	0.3469	0.3504	0.3623	0.3646	0.3598
0.3	0.3591	0.3683	0.3623	0.3607	0.3669	0.3656
0.35	0.3535	0.3493	0.3625	0.3610	0.3654	0.3665
0.4	0.3505	0.3506	0.3396	0.3636	0.3655	0.3524
0.45	0.3447	0.3643	0.3582	0.3576	0.3592	0.3516
0.5	0.3646	0.3533	0.3577	0.3575	0.3665	0.3664

Table 4.3: F1 score for hair masks based on the HNN output

	big threshold (α_{thres})					
small threshold (β_{thres})	0.5	0.55	0.6	0.65	0.7	0.75
0.1	0.7262	0.7183	0.7310	0.7395	0.7356	0.7144
0.15	0.7220	0.7303	0.7316	0.7296	0.7381	0.7338
0.2	0.7301	0.7299	0.7340	0.7334	0.7399	0.7493
0.25	0.7340	0.7355	0.7372	0.7435	0.7449	0.7423
0.3	0.7418	0.7466	0.7435	0.7426	0.7459	0.7426
0.35	0.7389	0.7368	0.7436	0.7428	0.7453	0.7457
0.4	0.7376	0.7374	0.7310	0.7442	0.7444	0.7383
0.45	0.7343	0.7445	0.7413	0.7410	0.7419	0.7378
0.5	0.7447	0.7387	0.7410	0.7410	0.7458	0.7465

As shown in Table 4.2 and Table 4.3, the best performance for the HNN hair mask occurs when small threshold (β_{thres}) is 0.2 and big threshold (α_{thres}) is 0.75. This shows that the optimal performance occurs when the big threshold (α_{thres}) is large and small threshold (β_{thres}) is at mid-range. Having a large big threshold (α_{thres}) is beneficial since this gives us more confidence when gathering the positive samples. We do not need to worry so much for not having enough positive samples since we added rotation when we gather positive samples (which greatly increases the variations in the positive samples). On the other hand, it is the best to have the small threshold (β_{thres}) at mid-range. This is due to the fact that when the small threshold (β_{thres}) is too big, many positive hair patches would be included as negative training samples, which can confuse the neural network. However, when the small threshold (β_{thres}) is too small, many negative training samples are not captured, and the neural network would not have seen enough variations in the negative samples (and thus not being able to classify many negative samples correctly). Comparing to the ModiFace [8] hair mask (performance score = 0.3632), the HNN hair mask has slightly better performance (performance score = 0.3734).

A closer look of the performance change as the threshold value (γ_{thres}) changes for the best performance thresholds for the HNN hair mask ($\alpha_{thres} = 0.75$, $\beta_{thres} = 0.2$) is shown in Fig 4.8 and Fig 4.9.

Comparing Fig 4.8 and Fig 4.9, the change in performance is more significant in performance scores than F1 scores as we vary threshold values (γ_{thres}). However, both performance measures follow a similar trend and peak at the same threshold value. The HNN hair mask performance peaks at a relatively

Figure 4.8: Performance scores for different thresholds (γ_{thres}) for HNN hair mask Figure 4.9: F1 scores for different thresholds (γ_{thres}) for HNN hair mask

large threshold ($\gamma_{thres} = 0.7$). This makes sense since there are naturally more non-hair patches than hair patches in a given image, so the neural network has its best performance when it is slightly bias toward the negative patches. The detailed performance at the best thresholds ($\gamma_{thres} = 0.7$) is shown in Table 4.4.

As shown in Table 4.4, the HNN hair mask has higher accuracy and lower over-estimation than the ModiFace [8] hair mask, leading to a boost in performance. At the same time, both precision and recall

Table 4.4: Best performance comparison for HNN hair mask

	accuracy	over-estimation	performance	precision	recall	F1
HNN	0.7301	0.2185	0.3734	0.7696	0.7301	0.7493
ModiFace [8]	0.7274	0.2281	0.3632	0.7613	0.7274	0.7439

increase for HNN hair mask, causing an increasing in the F1 score. Comparing to Table 4.1, the lower performance of the HNN output comparing to the result from training on the ground truth is mainly due to lower accuracy (for lower performance score) and lower recall (for lower F1 score). In other words, less ground truth is detected in this case comparing to that of the HNN trained using the ground truth.

Increasing the Number of Training Samples

Instead of using the minimal training set where we stop gathering the training samples when we have used up either the positive set or the negative set, we can increase the number of training samples with a maximal training set: keep getting training samples until we have used up both positive and negative sets. With a rotation step of 5 degrees, the performance score increased from 0.3734 to 0.4022 and the F1 score increased from 0.7493 to 0.7633 after the maximal training set was used, as shown in Table 4.5.

Table 4.5: Performance comparison for different number of training samples for HNN hair masks

training data set	accuracy	over-estimation	performance	precision	recall	F1
minimal training data	0.7301	0.2185	0.3734	0.7696	0.7301	0.7493
maximal training data	0.7677	0.2437	0.4022	0.7590	0.7677	0.7633

Changing the Number of Rotations

Instead of fixing the rotation step at 5 degrees, we experimented with different rotation steps with maximal training (include all positive training samples and negative training samples in the training set). The performance score and the F1 score is shown in Fig 4.10 and Fig 4.11, respectively.

As shown in Fig 4.10 and Fig 4.11, both the performance score and F1 score are higher when we include more rotations in the positive training samples. This makes sense since more training samples allow the neural network to see more variations in the positive samples and allows it to generalize better for unseen testing patches.

Increasing the Number of Hidden Layers

In this section, we explored HNN with two and three hidden layers. A deeper neural network has more capacity and is able to find more complex features. However, a deeper neural network is also easier to get stuck at poor local minima. Therefore, it is necessary to pre-train the weights of the neural network as described in Section 3.1.3. We experimented with maximal training with a rotation step of 1. The performance of HNN with more than one hidden layer is shown in Table 4.6.

Figure 4.10: Performance score for different rotation steps

Figure 4.11: F1 score for different rotation steps

As shown in Table 4.6, both HNN with 2 hidden layers and 3 hidden layers have a much lower performance when there is no pre-training than pre-training. This shows that pre-train the weights with a deep belief net is effective in avoiding poor local minima. However, the performance scores of the HNN with 2 and 3 hidden layers do not significantly improve from that of the 1-hidden layer HNN (performance score = 0.4022). Thus, adding more hidden layers in this case does not provide many additional benefits.

Table 4.6: Performance of multi-layer HNN

	no pre-training		pre-training	
number of hidden layers	performance	F1	performance	F1
2	0.3758	0.7504	0.4028	0.7633
3	0.3582	0.7423	0.4009	0.7624

Sample Output

Several sample HNN outputs are shown in Fig 4.12.

Figure 4.12: Sample outputs: First row: input image; Second row: ModiFace hair mask output [8]; Third row: HNN hair mask output

4.2.4 Combined Hair Mask

In this section, we try to explore the possibility of combining the HNN output with the ModiFace hair mask [8] described in section 2.2. To combine the HNN hair mask with the ModiFace hair mask [8], we

just multiply the output of the HNN with the color similarity score, the gradient similarity score and the prior probability for each patch in the hair mask before turning it into a binary mask.

The best performance score and F1 score of the combined mask is shown in Table 4.7.

Table 4.7: Best performance comparison for combined hair masks

	accuracy	over-estimation	performance score	precision	recall	F1
HNN hair mask	0.7677	0.2437	0.4022	0.7590	0.7677	0.7633
combined hair mask	0.7488	0.2061	0.4064	0.7842	0.7488	0.7661

As shown in Table 4.7, the performance of the HNN hair mask does not improve by very much after we combine with the ModiFace hair mask [8]. The reason for this is the HNN hair mask is trained on samples taken from the ModiFace hair mask [8]. Thus, the HNN hair mask covers most patterns seen in the ModiFace hair mask [8]. Combining the HNN hair mask with the heuristic hair mask is able to take advantage of both masks, but the improvement is very limited.

Sample Output

Several sample combined hair masks are shown in Fig 4.13.

4.2.5 Runtime

We performed all the experiments on a 2.7 GHz Intel i5 CPU. The runtime breakdown for minimal training and maximal training of the HNN hair mask with a 5 degrees rotation step is shown in Table 4.8. The bottleneck for both cases is during the positive sample generation stage. More specifically, generating rotated positive samples take the largest percentage of the total runtime in the both cases (around 50%). This is due to the fact that each time we rotate the image, we need to go through all the pixels in the image to extract the hair patches. At the same time, as we increase the number of training samples, the neural net training time increases significantly. If more layers are added to the neural net, the run-time difference in training the neural net will be more significant.

Table 4.8: Runtime breakdown for heuristically-trained neural network for a single image

	minimal training (sec)	maximal training (sec)
ModiFace [8]	40.3	40.3
positive sample generation	105.7	105.7
negative sample generation	2.1	2.1
training set generation	2.3	4.9
neural net training	7.8	75
mask creation	15.2	15.2
total	173.4	243.2

Figure 4.13: Sample outputs: First row: input image; Second row: ModiFace hair mask output [8]; Third row: HNN hair mask output; Fourth row: combined hair mask output

4.2.6 Results Comparison

In this section, the goal is to compare HNN with DNN and ModiFace hair mask [8]. In order to have a fair comparison of the three methods, we constructed a training set and a test set by dividing the images randomly into two sets (so that each set has 50 images). We constructed 10 of such sets. The final performance is the average performance of the 10 sets. Since we do not need training sets for the ModiFace hair mask [8] and HNN hair mask, we just ignore the training sets when constructing the hair masks (i.e. the performance for each set only depends on the test set).

The comparison results is shown in Table 4.9. We experimented with both the original ModiFace hair mask [8] method and version implemented for this thesis (used to take training samples). Both versions achieved very similar performance.

Table 4.9: Performance comparison of different methods

	Performance score	F1 score
DNN	0.1565	0.6316
ModiFace [8]	0.3721	0.7484
Implementation of ModiFace [8]	0.3659	0.7450
HNN	0.4058	0.7647

As shown in Table 4.9, HNN out-performs all other methods and it improves the performance of the ModiFace heuristic hair mask [8] by 9 %.

Fig 4.14 and Fig 4.15 show two examples that compares the output of the ModiFace hair mask [8] and the HNN hair mask. In each example, we first constructed an aggressive hair mask by using a smaller threshold for the ModiFace hair mask [8] (first column in each image). This hair mask gets a large portion of the ground truth, but also gets many errors. We then constructed a conservative hair mask by using a bigger threshold for the ModiFace hair mask [8] (second column in each image). This hair mask contains less errors, but misses a large portion of the ground truth. The last column in each image shows the HNN hair mask. It gets a large portion of the ground truth while not getting many errors. It is also able to detect some hair regions that were detected by neither the aggressive hair mask nor the conservative hair mask. This indicates that the HNN hair mask has learned something new from the training samples and is more effective in hair segmentation. The detailed performance comparison for the examples are shown in Table 4.10 and Table 4.11, respectively.

Table 4.10: Performance for Fig 4.14 comparison

	accuracy	over-estimation	performance	precision	recall	F1
small threshold	0.8777	0.3866	0.4310	0.6942	0.8777	0.7753
big threshold	0.5263	0.0272	0.2627	0.9508	0.5263	0.6776
HNN	0.8424	0.1602	0.5746	0.8402	0.8424	0.8413

Table 4.11: Performance for Fig 4.15 comparison

	accuracy	over-estimation	performance	precision	recall	F1
small threshold	0.9586	0.8811	0.0742	0.5210	0.9586	0.6751
big threshold	0.5236	0.1102	0.2165	0.8261	0.5236	0.6410
HNN	0.9609	0.1709	0.7590	0.8490	0.9609	0.9015

Figure 4.14: First sample comparison between the ModiFace hair mask [8] and the HNN output. First column shows the result of using an aggressive ModiFace hair mask [8]; the second column shows the result of using a conservative ModiFace hair mask [8]; the third column shows the HNN output

Figure 4.15: Second sample comparison between the ModiFace hair mask [8] and the HNN output. First column shows the result of using an aggressive ModiFace hair mask [8]; the second column shows the result of using a conservative ModiFace hair mask [8]; the third column shows the HNN output

4.2.7 Discussion

We experimented with a heuristically-trained neural network in which we utilized human-defined heuristics to take training samples in each image. The human-defined heuristics cannot guarantee 100% that a certain patch in the input image is positive or negative, but it gives a confidence measure for each patch. We then use this information to select the positive and negative samples to train a neural network for each image.

Comparing to the universal neural network described in chapter 3, HNN has much better performance. However, since we need to train a neural network for each image, the testing time is much longer. This method also has better performance than the ModiFace [8] method mentioned in Section 2.2.

In the future, HNN can be further optimized (e.g. to be implemented on GPU) so that better runtime can be achieved.

Chapter 5

Conclusions

In this thesis, we present a method for binary classification using neural networks that performs training and classification on the same data using the help of a pre-training heuristic classifier. The heuristic classifier is initially used to segment data into three clusters of high confidence positives, high confidence negatives, and low confidence sets. The high confidence sets are used to train a neural network (NN) which is then used to classify the low confidence set. Applying this method to the binary classification of hair vs. non-hair patches, we obtain a 9% increase using the heuristically-trained NN over the current state of the art hair segmentation method.

In the meanwhile, we evaluated the distribution of gradient directions in hair patches, and explored the relation between proximity to the face and the angle of the gradients. We quantitatively evaluated the role of texture in hair patches, which illustrates what can be learned and applied by machine learning systems for texture-based hair detection.

In addition to the heuristically-trained neural networks, two other hair segmentation methods were implemented to serve as a point of comparison. The first method is a heuristic-based method that segments hair based on color and textural information. It first captures potential hair samples in a given image and then constructs a color model and a gradient model based on the hair samples. To perform segmentation, each patch is compared with the hair color and the gradient model, and is labelled as a hair patch if it has both similar color and texture as the hair models. However, this method is very sensitive to the different threshold values, and thus has large performance variances between different images. The second method is a deep neural network (DNN). In this case, we first extract many hair patches and non-hair patches from all images in the training set and use them to train the DNN, aiming to let the DNN learn patterns of the hair and non-hair patches. After the DNN is trained, we then use it to classify images in the testing test. However, this method did not perform very well since we did not have a large number of training images. When comparing the performance of the HNN with the heuristics-based hair mask and DNN hair mask, HNN out-performs both methods.

There are many aspects of this thesis that can be improved in the future. First of all, the gradient score in the heuristic hair mask can be normalized, which will result in a more accurate gradient magnitude score. However, more time will be devoted into finding the best threshold that turns the probability

mask into a binary mask as the optimal threshold will be very small. Additionally, gradient direction information can also be used in the heuristic hair mask since we have a rough idea of the hair directions at different locations from the hair texture analysis.

Furthermore, the performance of the DNN will be greatly improved if more training images are used to train it. Therefore, in the future, we can expand the database and experiment with DNN again to see if the performance can be improved.

Moreover, we can increase the capacity of the HNN by making it deeper (adding more hidden layers to HNN). In theory, a more complex HNN should have even better performance since it is able to find more complex features. However, a deeper HNN is more difficult to train (more time needs to be spend on finding the optimal learning parameters). Another issue before we can make HNN deeper is run-time. The current run-time is very slow already, and adding more layers to HNN will make the run-time even slower. Thus, in the future, we should improve the run-time by optimizing the code with parallel programming or implement the model on a GPU.

Bibliography

- [1] H. Chen and S.-C. Zhu, "A generative sketch model for human hair analysis and synthesis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, Jul. 2006.
- [2] D. Patrick and S. Bangay, "A Lightwave 3D Plug-In for Modeling Long Hair on Virtual Humans," *Proc. Second Intl Conf. Comput. Graph., Virtual Reality, Visualisation and Interaction in Africa*, pp. 161-187, 2003.
- [3] M. Koster, J. Haber, and H.-P. Seidel, "Real-Time Rendering of Human Hair Using Programmable Graphics Hardware," *Proc. Comput. Graph. Intl (CGI 04)*, pp. 248-256, June 2004.
- [4] B. Choe, M. Choi, and H.-S. Ko, "Simulating Complex Hair with Robust Collision Handling," *Proc. ACM SIGGRAPH/Eurographics Symp. Comput. Animation (SCA 05)*, pp. 153-160, 2005.
- [5] M. Li, S. Bao, W. Dong, Y. Wang, and Z. Su, "Head-shoulder based gender recognition," *Proc. 20th IEEE Intl Conf. (ICIP '13)*, pp. 2753 - 2756, Sept. 2013.
- [6] F. Min, Z. Kun, and N. Sang, "A Classified Method of Human Hair for Hair Sketching," *Congr. on Image and Signal Process. (CISP '08)*, vol. 4, pp. 109 - 114, May 2008.
- [7] C. Rousset and P. Y. Coulon, "Frequential and color analysis for hair mask segmentation," *Proc. 15th IEEE Intl Conf. (ICIP '08)*, pp. 2276-2279, Oct. 2008.
- [8] P. Aarabi, "Automatic Segmentation of Hair In Images," *Proc. IEEE Intl Symposium on Multimedia*, 2015 (to appear).
- [9] K. Chuang, et al. "Fuzzy c-means clustering with spatial information for image segmentation," *Computerized Medical Imaging and Graphics*, vol. 30, no. 1, Jan. 2006.
- [10] W. Cai, S. Chen, and D. Zhang, "Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation," *Pattern Recognition*, vol. 40, no. 3, Mar. 2007.
- [11] Z. Yu, H. Wong, and G. Wen, "A modified support vector machine and its application to image segmentation," *Image and Vision Computing*, vol. 29, no. 1, Jan. 2011.
- [12] X. Wang, T. Wang, J. Bu, "Color image segmentation using pixel wise support vector machine classification," *Pattern Recognition*, vol. 44, no. 4, Apr. 2011.

- [13] P. Salembier, P. and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Trans. Image Processing*, vol. 9, no. 4, Apr. 2000.
- [14] P. Yin, A. Criminisi, J. Winn, and I. Essa, "Tree-based Classifiers for Bilayer Video Segmentation," *Proc. IEEE Conf. (CVPR '07)*, pp.1-8, Jun. 2007.
- [15] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," *Proc. NIPS 2012*, 2012.
- [16] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks." *Proc. Advances in Neural Information Processing Systems*, pp. 1090-1098, 2012.
- [17] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, Nov. 1958.
- [18] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the microstructure of Cognition*, Vol. I, Cambridge, MA, MIT Press, 1986, pp. 319-362.
- [19] D. Erhan, et al. "Why Does Unsupervised Pre-training Help Deep Learning?," *Journal of Machine Learning Research*, vol. 11, Feb. 2010.
- [20] J. Tompson, et al. "Joint training of a convolutional network and a graphical model for human pose estimation." *Proc. Advances in Neural Information Processing Systems*, pp. 1799-1807, 2014.
- [21] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," *Proc. IEEE Intl Conf. (ICASSP 2013)*, pp. 6645 - 6649, 2013.
- [22] G. Hinton, et al. "Deep neural networks for acoustic modeling in speech recognition: The Shared Views of Four Research Groups ," *IEEE Signal Processing Magazine*, vol. 29, no. 6, Oct. 2012.
- [23] R. Sarikaya, G. Hinton, and A. Deoras, "Application of Deep Belief Networks for Natural Language Understanding," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 22, no. 4, Feb. 2014.
- [24] R. Sarikaya, and Hinton, G. "Deep belief nets for natural language call-routing ," *Proc. IEEE Intl Conf. (ICASSP 2011)*, pp. 5680 - 5683 , 2011.
- [25] Y. Bengio, "Introduction," in *Learning Deep Architectures for AI*, Hanover, MA, USA: Now Publishers Inc, 2009, ch. 1, sec. 1-3, pp. 1-9.
- [26] T. Serre, G. Kreiman, M. Kouh, C. Cadieu, U. Knoblich, and T. Poggio, "A quantitative theory of immediate visual recognition," *Progress in Brain Research, Computational Neuroscience: Theoretical Insights into Brain Function*, vol. 165, pp. 335-362, 2007.
- [27] D. Wang, et al. "A novel two-tier Bayesian based method for hair segmentation," *Proc. 16th IEEE Intl Conf. (ICIP '09)*, pp. 2401-2404, Nov. 2009.
- [28] D. Wang et al. "A novel coarse-to-fine hair segmentation method ," *Proc. IEEE Intl Conf. on Automatic Face and Gesture Recognition (FG 2011)*, pp. 233 - 238, Mar. 2011.

- [29] D. Wang et al. "Isomorphic Manifold Inference for hair segmentation," *Proc. 10th IEEE Intl Conf. and Workshps on Automatic Face and Gesture Recognition (FG)*, pp. 1 - 6, Apr. 2013.
- [30] K-C Lee, D. Anguelov, B. Sumengen, and B. Gokturk, "Markov Random Field Models for Hair and Face Segmentation, *Proc. 8th IEEE Intl Conf. (FG '08.)*, pp. 1-6, Sept. 2008.
- [31] N. Wang, H. Ai, and S. Lao, "A Compositional Exemplar-Based Model for Hair Segmentation," *Proc. 10th Asian Conf. on Computer Vision*, pp. 171-184, Nov. 2011.
- [32] P. Julian, et al. "Automatic hair detection in the wild, *Proc. 20th Intl Conf. on Pattern Recognition (ICPR)*, pp. 4617 - 4620 , Aug. 2010.
- [33] B. Manashirov and P. Aarabi, "Learning from 1,000,000 user-uploaded faces," *Proc. IEEE Intl Conf. on Multimedia and Expo*, pp. 1342 - 1345 , Jul. 2009.
- [34] W. S. Mokrzycki and M. Tatol, "Color difference Delta E - A survey," *Machine Graphic & Vision*, vol. 20, no. 4, Apr. 2011.
- [35] I. Sobel, "Camera models and machine perception," Ph.D. dissertation, Stanford Univ., Stanford, CA, 1970.
- [36] W. Burger and M. J. Burge, "Filters," in *Digital Image Processing - An Algorithmic Introduction Using Java*, New York, NY, USA: Springer Science & Business Media, 2012, ch. 6, sec. 2.7, pp. 96-98.
- [37] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks: a review," *Pattern Recognition*, vol. 35, no. 10, Oct. 2002.
- [38] R. Lienhart and A. Wernicke, "Localizing and segmenting text in images and videos," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 4, Apr. 2002.
- [39] S. M. Bhandarkar, J. Koh, and M. Suk, "Multiscale image segmentation using a hierarchical self-organizing map," *Neurocomputing*, vol. 14, no. 3, Feb. 1997.
- [40] W.E. Reddick, et al "Automated segmentation and classification of multispectral magnetic resonance images of brain using artificial neural networks," *IEEE Trans. Medical Imaging*, vol. 16, no. 6, Dec. 1997.
- [41] R. Girshick, et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Proc. IEEE Conf. (CVPR '14)*, pp.580 - 587, Jun. 2014.
- [42] M. Egmont-Petersen and E. Pelikan, "Detection of Bone Tumours in Radiographic Images using Neural Network," *Pattern Analysis & Applications*, vol. 2, no. 2, Jun. 1999.
- [43] L. Zhao and C.E. Thorpe, "Stereo- and neural network-based pedestrian detection," *IEEE Trans. Intelligent Transportation Systems*, vol. 1, no. 3, Sep. 2000.
- [44] W. Guo and P. Aarabi, "Quantitative Evaluation of Hair Texture," *Proc. IEEE Intl Symposium on Multimedia*, 2015 (to appear).

- [45] C. Loizou, V. Murray, M. S. Pattichis, I. Seimenis, M. Pantziaris, and C. S. Pattichis, "Multi-scale amplitude modulation-frequency modulation (AM-FM) texture analysis of multiple sclerosis in brain MRI images," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 1, Jan. 2011.
- [46] Z. Karimaghloo, H. Rivaz, D. L. Arnold, D. L. Collins, and T. Arbel, "Temporal Hierarchical Adaptive Texture CRF for Automatic Detection of Gadolinium-Enhancing Multiple Sclerosis Lesions in Brain MRI," *IEEE Trans. Medical Imaging*, vol. 34, no. 6, Dec. 2014.
- [47] Z. I. Petrou, I. Manakos, T. Stathaki, C. A. Mcher, and M. Adamo, "Discrimination of Vegetation Height Categories With Passive Satellite Sensor Imagery Using Texture Analysis," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 4, Mar. 2015.
- [48] I. Champion, C. Germain, J. P. D. Costa, A. Alborini, and P. Dubois-Fernandez, "Retrieval of Forest Stand Age From SAR Image Texture for Varying Distance and Orientation Values of the Gray Level Co-Occurrence Matrix," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 1, Mar. 2013.
- [49] X. Mao, M. Kikukawa, K. Kashio and A. Imamiya, "Automatic Generation of Hair Texture with Line Integral Convolution," *Proc. IEEE Intl Conf. on Inform. Visualization*, pp. 303-308, Jul. 2000.
- [50] F. Cheraghchi and W. Lee, "Hair Strand Extraction by Template Matching Using Gabor Filter and Histogram," *4th Intl Conf. (IPTA)*, pp. 1-6, Oct. 2014.
- [51] G. Dong and M. Xie, "Color clustering and learning for image segmentation based on neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 925-936, Jul. 2005.
- [52] L. Maddalena and A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1168-1177, Jul. 2008.
- [53] Z. Zhang, Q. Zhu and Y. Xie, "Learning based alpha matting using support vector regression," *Proc. 19th IEEE Intl Conf. (ICIP '13)*, pp. 2109 - 2112, Sept. 2012.
- [54] S. Y. Kung, "Linear regression and discriminant analysis for supervised classification," in *Kernel Methods and Machine Learning*, Cambridge, United Kingdom: Cambridge University Press, 2014, ch. 8, sec. 2, pp. 249-252.
- [55] E. Alpaydin, "Supervised Learning," in *Introduction to Machine Learning*, 3rd ed., Cambridge, MA, USA: MIT Press, 2014, ch. 2, sec. 7, pp. 37-40.
- [56] G. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, Jul. 2006.
- [57] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," <http://arxiv.org/abs/1207.0580>, 2012.
- [58] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and F-score, with implication for evaluation," *ECIR'05 Proc. of the 27th European Conf. on Advances in Inform. Retrieval Research*, pp. 345-359, 2005.
- [59] D.D. Lewis and W.A. Gale, "A sequential algorithm for training text classifiers," *Proc. 17th annual Intl ACM SIGIR Conf. on Research and Development in Information Retrieval*, pp. 3-16, 1994.

- [60] S. Tong and E. Chang, “Support vector machine active learning for image retrieval,” *Proc. 9th ACM Intl Conf. on Multimedia*, pp. 107-118, 2001.
- [61] A. Bordes, et al. “Fast Kernel Classifiers with Online and Active Learning,” *The Journal of Machine Learning Research*, vol. 6, Dec. 2005.
- [62] H. T. Nguyen and A.Smeulders, “Active learning using pre-clustering,” *Proc. 21st Intl Conf. on Machine learning* , pp. 79-87, 2004.