



DEGREE PROJECT IN INFORMATION AND COMMUNICATION  
TECHNOLOGY,  
SECOND CYCLE, 30 CREDITS  
*STOCKHOLM, SWEDEN 2019*

# **Measure face similarity based on deep learning**

**CHENYANG ZHOU**



# **Measure face similarity based on deep learning**

CHENYANG ZHOU

Master in Embedded Systems  
Supervisor: Mårten Björkman  
Examiner: Danica Kragic Jensfelt  
School of Electrical Engineering and Computer Science  
Principal: KSTING AB  
Supervisor in principal: Erwan Lemonnier  
Email: czho@kth.se  
Date: May 17, 2019



## **Abstract**

Measuring face similarity is a task in computer vision that is different from face recognition. It aims to find an embedding in which similar faces have a smaller distance than dissimilar ones. This project investigates two different Siamese networks to explore whether these specific networks outperform face recognition methods on face similarity. The best accuracy is from a Siamese convolution neural network, which is 65.11%. Moreover, the best results in a similarity ranking task are obtained from Siamese geometry-aware metric learning. Besides, this project creates a novel dataset with facial image pairs for face similarity.

## **Sammanfattning**

### **Mätning av ansiktslikhet baserad på djupinlärning**

Mätning av ansiktslikhet är en uppgift i datorseende som skiljer sig från ansiktsigenkänning. Det syftar till att hitta en inbäddning där liknande ansikten har ett mindre avstånd än olika ansikten. Detta projekt undersöker två olika siamesiska nätverk för att utforska om dessa specifika nätverk överträffar ansiktsigenkänningsmetoder på ansiktslikhet. Den bästa noggrannheten är från ett Siamesiskt faltningsnätverk, vilket är 65,11%. Dessutom erhålls de bästa resultaten i en likhetsrankningsuppgift från Siamesisk geometri-medveten metrisk inlärning. Projektet skapar också ett nytt dataset med ansiktsbildpar för ansiktslikhet.

## Acknowledgment

I want to express my gratitude to all the people who helped me with this project and thesis in some way.

Firstly, I want to thank Mårten Björkman for being my supervisor. I am very grateful for his guidance and expertise that contributed a lot to this project. Also, he is the one who introduced me to computer vision through his course in KTH, which is the biggest reason I chose this field as a degree project. I also want to thank Danica Kragic Jensfelt for being my examiner.

Secondly, I would like to thank my industrial supervisor Erwan Lemonnier, for giving me the opportunity to conduct this project. I am very grateful for his support in providing the platform and cloud storage, which save much time during the project.

Thirdly, I want to thank my one of my best friends Xunyu Zuo, who accompanied me during the project. I am very grateful for his making my time joyful when I felt boring and helpless.

Finally, I want to thank my beloved parents. I am very grateful for their support and encouragement. Their love is the biggest power for me to overcome difficulties in my life.

# Contents

<b>Chapter 1 Introduction</b> .....	1
1.1 Background.....	1
1.2 Problem statement.....	2
1.3 Contribution.....	3
1.4 Limitation.....	4
1.5 Ethical and societal aspects .....	5
1.6 Outline.....	5
<b>Chapter 2 Theoretical background</b> .....	6
2.1 Machine learning and artificial neural network.....	6
2.1.1 Neuron and activation function.....	7
2.1.2 Architecture and fully connected .....	8
2.1.3 Segmentation and overfitting.....	10
2.1.4 Loss function and optimization .....	10
2.2 Convolutional neural network.....	14
2.2.1 Convolutional layer .....	14
2.2.2 Pooling layer.....	16
2.3 Metric learning .....	17
2.3.1 Siamese network.....	18
2.3.2 Contrastive loss .....	19
<b>Chapter 3 Related work</b> .....	20
3.1 Face recognition.....	20
3.1.1 Different methods for face recognition .....	21
3.1.2 VGG-Face.....	23
3.1.3 The difference between face recognition and similarity .....	23
3.2 Image similarity.....	25



<b>Chapter 4 Methods</b> .....	27
4.1 Data collection.....	27
4.2 Experiment approaches.....	32
4.2.1 Experiment 1: Measuring similarity based on CNN ..	32
4.2.2 Experiment 2: Measuring similarity based on GAML	35
4.3 Experiment setting .....	37
4.3.1 Dataset segmentation .....	37
4.3.2 Training parameters and protocol .....	38
4.4 Evaluation metrics.....	39
<b>Chapter 5 Result and analysis</b> .....	41
5.1 Prediction result and analysis .....	41
5.2 Similarity ranking result.....	43
<b>Chapter 6 Conclusion</b> .....	47
<b>Chapter 7 Future work</b> .....	48
<b>Bibliography</b> .....	49



# Chapter 1

## Introduction

This thesis is divided in 6 main chapters: introduction, theory, related work, methods, results, and discussion followed by conclusion and future work. In this chapter, there is an overall introduction of this degree project, including background motivation, problem domain statement, main contribution, limitation, and the outline of this report.

### 1.1 Background

Faces are unique for everybody, and it is a direct and visible sign for us to identify each other in our daily life. Sometimes we might find that a person looks like another one we know, even if they do not have a kinship. It is not difficult for us to recognize them as two identities, but there are still some common characteristics between them, which make us think they look similar. Although we might have no idea how to describe these common characteristics, the powerful processing system in our brain can deal with them quickly and mark those two persons as similar.

Similar faces play an essential role in some fields, like the movie industry and model industry. In the movie industry, dangerous and complicated stunts are one of the most attractive elements in an action movie. However, it is sometimes difficult for a famous actor to perform these stunts perfectly by himself. In this case, a producer might choose

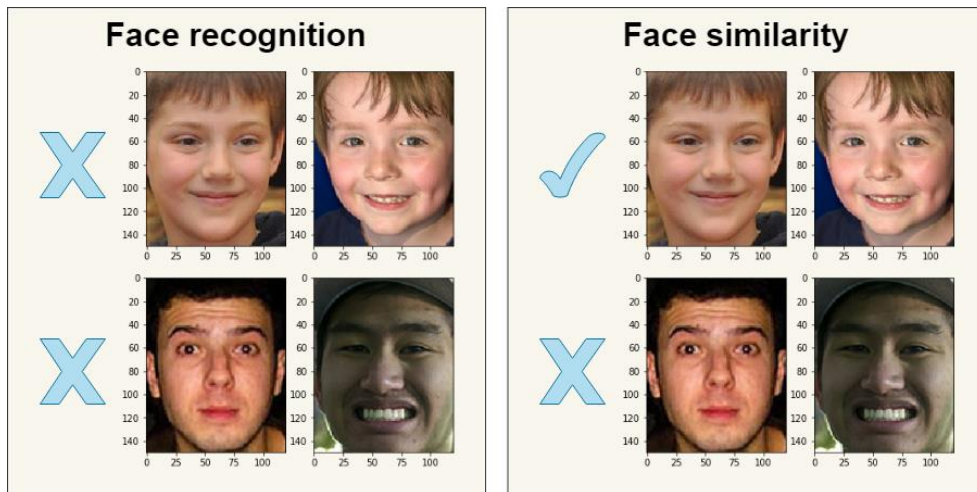
a skillful stuntman to complete this performance. If this stuntman has a similar face with the actor, the audience will think intuitively that this uninterrupted action is performed by the same actor. In the model industry, there is usually a theme in a fashion show or magazine photography. The organizer or designer might prefer to choose models with some special characteristics, like a wide forehead and high cheekbones, to match the theme. So when the organizer contacts agencies to find proper models, he might choose a series of candidates at first and screen them later. If the system can list models automatically that have similar faces with a chosen one, it could save much time from selecting them one by one.

## **1.2 Problem statement**

Face recognition consists of two main tasks: face identification (find the identity from a facial image) and face verification (verify whether two faces belong to the same person). They are both familiar and classic tasks in the field of computer vision and have been tackled with great success after deep learning became popular and implementable [3, 18]. The algorithms or networks used for face recognition are trained with identity as categories, and most of them try to find an embedding space where the distance between different persons is large while that between the same person is small.

Different from face recognition, this degree project focuses on face similarity. As shown in Figure 1.1, it is a new task to verify whether two faces look similar or not. Researchers have started to pay more attention to face similarity [1, 2] in recent years. They tried to develop algorithms to map faces into an embedding space in which the distance between similar pairs is small while that between dissimilar pairs is large. However, the label of an image pair in face similarity is dependent on subjective marking, which is variable among different people. Since the face similarity task is highly related to people's individual choices, it is meaningful to implement face similarity tasks based on machine learning to explore how people think when they judge whether two faces are similar or not.

The primary purpose of this degree project is to develop a method to measure face similarity between different identities. This method is a specific function which outperforms other face recognition algorithms on face similarity. A good choice to implement this function is to use machine learning since it is practical to learn the discrimination between visual similarity and dissimilarity with machine learning and there is no need to design a specific descriptor manually. The research problem devised from this purpose is concentrated on the following question: To what extent can the face similarity function discriminate the similarity and dissimilarity of image pairs based on machine learning?



**Figure 1.1** The difference between face recognition and face similarity. Face recognition algorithms try to enlarge the distance between faces from different identities while face similarity algorithms attempt to narrow the distance between similar faces. In the left figure, face recognition algorithm predicts these two pairs of images as different identities regardless of whether they look similar. In the right figure, face similarity algorithm predicts each pair according to whether they are visually similar.

### 1.3 Contribution

One of the contributions is a novel dataset containing ground-truths

of similar and dissimilar facial image pairs. Since there is no available dataset collected according to face similarity between different identities at present, a novel dataset is collected which contains plenty of face pairs with labels which are annotated by human individuals. Also, each pair of facial images is labeled by three people to reduce the impact of subjective judgment on similarity.

Another contribution are two face similarity functions: one of them is developed based on existing face recognition descriptor, and the other one is implemented with a convolutional neural network. Both of them can map the original facial image into a specific similarity embedding space in which the distance between similar pairs is small and that between dissimilar pairs is large. So the similarity of images pairs can be scaled by the distance in the embedding space. There is also a comparison between these two functions to evaluate which one is more appropriate for face similarity task.

## **1.4 Limitation**

The first one is that all the evaluation of results is implemented on a novel dataset since there is no available dataset containing the label of similarity or dissimilarity. The following application of face similar function is not included in this thesis, which is a model scouting system to find models with similar faces. This application will use a dataset containing various model front faces, but there are no labels about the similarity in it. So the final results of the application will be evaluated by developers and guests from the principal.

The second one is that the alignment of faces is not included in this degree project. The raw images used to build the novel dataset for the face similarity task is the Names 100 dataset [19], in which all the facial images are collected on the website and aligned into  $120 \times 150$  pixels in advance. Most of these images are focused on the interesting part, the front face, in computer vision and remove those irrelevant, like hair and clothes, so there is no need to align them further.

## 1.5 Ethical and societal aspects

Even though many fields can benefit from the development of face similarity as introduced in Section 1.1, the task of similarity measurement raises some ethical and societal concerns at the same time. One problem is whether those images in the datasets are collected with the permissions from their owners. Another problem is whether those people in the datasets know that their pictures are used for scientific studies. It is essential to ensure that their privacy is not violated. Furthermore, the similarity features extracted from images are part of people's private information. The applications implementing face similarity measurement should protect private data at the same time.

## 1.6 Outline

The following part of this thesis consists of 6 other chapters.

- Chapter 2 introduces related theory used in this project, including deep learning and metric learning.
- Chapter 3 presents some state of the art researches, which are helpful for this project.
- Chapter 4 is a detailed description and illustration of the whole method used in this project, which contains data collection and network training. Experiment setting and evaluation methods are also included in this chapter.
- Chapter 5 is the experimental results and analysis. It also contains a comparison of two machine learning functions developed in this project.
- Chapter 6 gives the conclusion that is an answer to the stated problem in the first chapter.
- Chapter 7 presents the possible future work that might improve the face similarity function further.

# Chapter 2

## Theoretical background

This chapter presents some scientific theories involved in the project to enhance the comprehension of this thesis. There are three sections included: The first section provides an introduction regarding machine learning and neural networks. The second and third sections are based on the first one and give a more detailed description of two different learning models: deep learning and metric learning. These theories can help to understand the following sections more efficiently, especially related work in Section 3 and models in Section 4.

### 2.1 Machine learning and artificial neural network

Machine learning is a software technology to learn potential patterns buried in data rather than instructions from an engineer. For example, to accomplish a task which aims to find cats in images, machine learning can find a typical pattern in millions of images of cats, and this procedure is called training. Even though the information about the appearance of a cat is not provided in advance, the model after training can determine whether there is a cat in a new image by itself.

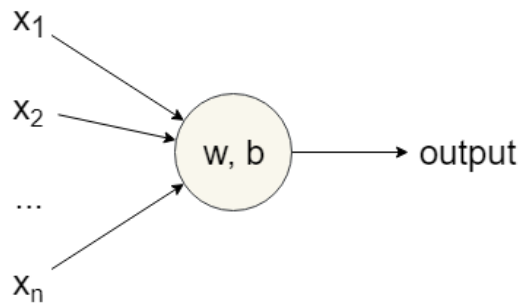
Artificial neural networks (ANNs) are a special type of machine learning algorithm, which is a model imitating neural networks in biology. It is widely used for prediction and classification nowadays



[30]. This section only provides some useful theories and algorithms used in this project. For further study, [30] is good material available on the internet.

### 2.1.1 Neuron and activation function

A neuron is the base unit of a neural network. The most simple neuron is called perceptron, which contains several binary inputs and a binary output, as shown in Figure 2.1. Each neuron has its weights and threshold.



**Figure 2.1** A simple neuron. It consists of several binary inputs:  $x_1, x_2, \dots, x_n$  and a binary output  $y$ . The output is computed by the weights and threshold in the neuron combined with the inputs.

The output is decided by the relationship of the weighted sum  $\sum_j w_j x_j$  and the threshold. If the sum is greater than the threshold, then the output is 1, and if the sum is less than the threshold, then the output is 0. To simplify the expression, the sum is changed into a dot product of two vectors:  $\mathbf{w} \cdot \mathbf{x} = \sum_j w_j x_j$ . Also, the threshold is changed into a bias:  $b = -threshold$ . Then the output is decided by  $\mathbf{w} \cdot \mathbf{x} + b$ . The criterion is formulated as follow:

$$output = \begin{cases} 0, & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases} \quad (2.1)$$

A small change of the weights or bias could result in a big flip of the output, from 0 to 1 or vice versa. Then the rest of the network may

change complicatedly beyond expectation. In this case, it is hard to observe how these parameters impact the output of the network. So activation functions are introduced to overcome this problem. The activation function in a neuron is applied to  $\mathbf{w} \cdot \mathbf{x} + b$ , and the output of a neuron is the result of the activation function. The criterion is:

$$\text{output} = \text{Activate}(\mathbf{w} \cdot \mathbf{x} + b) \quad (2.2)$$

Some common activation functions are listed in Table 1. We can find that a small change in weights and bias in a neuron results in a small change in the output. Moreover, the rate of change is decided by the first derivative of activation function.

**Table 1** Common activation functions

Name	Equation	First derivative
Logistic (or sigmoid)	$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh	$f(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$

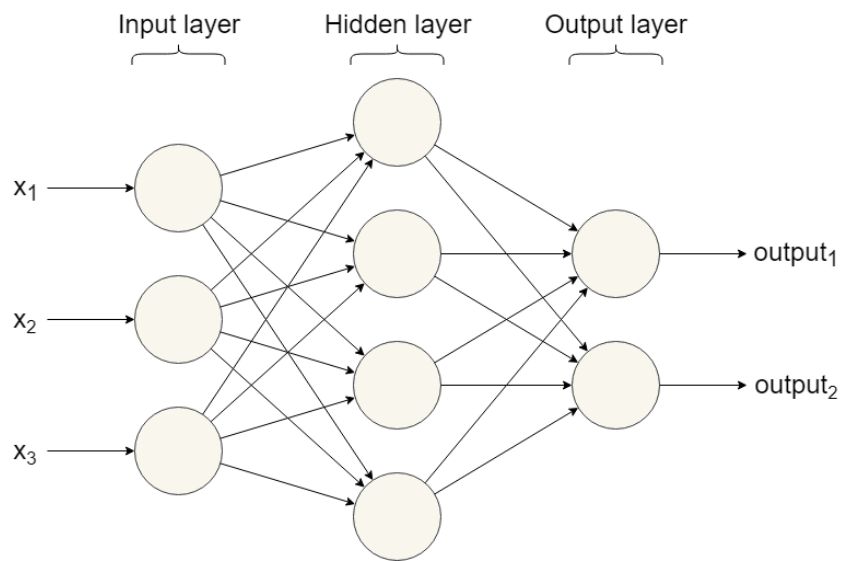
### 2.1.2 Architecture and fully connected

In most cases, a neural network consists of one input layer, one or more hidden layers, and one output layer. As shown in Figure 2.2, the input layer is the leftmost layer of the network, and the output layer is the rightmost one. The middle layers, which receive the output of the preceding layer as their input and deliver their output to the next layer, are called hidden layers. Each layer is composed of a series of neurons with various weights and biases. Suppose the  $i$ -th neuron in one layer holds the following equation:

$$output_i = \mathit{Activate} \left( \sum_j w_{ij} x_j + b_i \right) = \mathit{Activate}(\mathbf{w}_i \cdot \mathbf{x} + b_i) \quad (2.3)$$

Then, we get a weight matrix  $\mathbf{W}$ , with elements given by  $w_{ij}$  and a bias vector  $\mathbf{b}$  for each layer, and the relationship between input vector  $\mathbf{x}$  and output vector can be formulated as follow:

$$\mathbf{output} = \mathit{Activate}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (2.4)$$



**Figure 2.2** A simple fully connected ANN. It consists of one input layer with three neurons, one hidden layer with four neurons, and one output layer with two neurons.

When the data to be processed in a problem is complex and the classification results are multiple, the architecture of a neural network might become deeper and more complicated [30]. Furthermore, if all of the neurons are connected between two adjacent layers, then this network is fully connected. Figure 2.2 is also an example of a fully connected network.

### 2.1.3 Segmentation and overfitting

The whole dataset can be segmented into three different data sets to build a final neural network: training set, test set, and validation set. The training set contains samples that are used to fit the parameters of the network, i.e. the weights and biases. It usually consists of pairs of an input vector and an output target. The test set provides samples for evaluation of the final network. Samples in the test set should never be used in the training phase. Sometimes the loss, which will be explained in the next section, of the training set is small while that of the test set is large when overfitting occurs [31]. The validation set can provide samples in the training phase to evaluate the loss and monitor if the model has a good generalization.

### 2.1.4 Loss function and optimization

To obtain accurate prediction results from a network as much as possible, the parameters of the network are updated correspondingly. Each iteration of the training phase contains two procedures: forward propagation and backpropagation. The predicted results are computed by input values and parameters layer by layer in forwarding propagation and parameters are updated according to the difference between predicted and actual values in backpropagation.

A loss function (or cost function) is the function used during the training phase to measure the general difference between predicted results and ground-truths. The purpose of training is to minimize the loss value of the network. So it is crucial to find an appropriate loss function for different problems [30]. There are many loss functions applied to neural networks, and some standard functions are introduced in the following paragraphs. Other loss functions applied in this project are discussed in Section 2.3 and 3.2.

#### *L<sup>2</sup> loss function*

L<sup>2</sup> loss, which is also called mean square error (MSE), is a typical loss function in regression problems [31]. If we use  $y_i$  and  $\hat{y}_i$  to represent the  $i$ -th ground-truth and prediction output from the network respectively, the L<sup>2</sup> loss function can be formulated as Equation 2.5,

where  $N$  denotes the length of output vector.

$$\mathcal{L}_{L^2} = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2 \quad (2.5)$$

### *Cross-entropy loss function*

Cross-entropy loss is a normal loss function in classification problems [31]. It is used to measure the distance between two probability distributions of ground-truth and prediction. To transform the prediction output into its probability distribution between 0 and 1, the common method is to add a softmax layer at the end of a neural network. The softmax function is written as Equation 2.6, where  $P(\hat{y}_i)$  denotes the probability of  $i$ -th output  $\hat{y}_i$  and  $N$  denotes the length of output vector.

$$P(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_{j=1}^N e^{\hat{y}_j}} \quad (2.6)$$

Then we can get the cross-entropy loss formulated as Equation 2.7. The loss increases when the prediction results are different from the ground-truths [31].

$$\mathcal{L}_{cross-entropy} = - \sum_{i=1}^N y_i \log(P(\hat{y}_i)) \quad (2.7)$$

The procedure of minimizing the loss during the training phase is called optimization. This is implemented by some specific algorithms. The purpose of optimization is to change the parameters in the direction which is opposite to the gradient direction of loss function [31]. This method is called gradient descent, which can be formulated as follow:

$$\theta_{i+1} = \theta_i - \nu \nabla_{\theta_i} \mathcal{L}(\theta) \quad (2.8)$$

In the equation,  $\theta_i$  and  $\theta_{i+1}$  denote parameters of the  $i$ -th and  $(i + 1)$ -th iteration, respectively, and  $\nabla_{\theta_i} \mathcal{L}(\theta)$  denotes the gradient of loss function  $\mathcal{L}(\theta)$  when  $\theta = \theta_i$ . Learning rate  $\nu$  controls the speed of adjustment of parameters. If the learning rate is too small, it will take a long time to converge, while if it is too big, the loss function will fluctuate near the local minimum [31]. Common optimization strategies are presented in the following paragraphs.

### *Stochastic gradient descent*

Stochastic gradient descent (SGD) is an optimizer in which the parameters are updated once for each training sample, using randomly shuffled samples [31]. Even though it is more efficient to select one training sample randomly than to use the whole training set in an iteration, the final parameters are not always globally optimal. Mini-batch gradient descent (MBGD) is a tradeoff between efficiency and robustness. It selects a batch of samples randomly from the training set and updates parameters based on the samples in the batch for each iteration. The procedure of MBGD optimizer is summarized in Algorithm 1 based on [31].

---

**Algorithm 1** Mini-batch gradient descent (MBGD) optimizer based on [31]

---

**Input:** Training set  $\{x, y\}$ , batch size  $b$ , learning rate  $\nu$

**Output:** Parameters  $\theta$

Initialization: parameters  $\theta$

**for**  $iter = 1, 2, 3, \dots$  **do**

    Randomly select a mini-batch from the training set  $\{x^{(1)}, \dots, x^{(b)}\}$ ,  
    with corresponding targets  $y^{(i)}$

    Compute corresponding output of network  $\hat{y}^{(i)} = f(x^{(i)})$

    Compute gradient estimate:  $\hat{g} = \frac{1}{b} \nabla_{\theta} \sum_i \mathcal{L}(\hat{y}^{(i)}, y^{(i)}; \theta)$

    Update parameters  $\theta$  with  $\theta \leftarrow \theta - \nu \hat{g}$

**if**  $\theta$  converged **then**

        break

**end if**

**end for**

---

### *Adaptive momentum estimation*

Adaptive momentum estimation (ADAM) is one of the recent stochastic optimizers, which was proposed by Kingma, D. P. et al. in 2014 [32]. It updates the parameters according to both the average of gradient and the second moment of the gradient. The procedure of ADAM optimizer is summarized in Algorithm 2 based on [32].

---

**Algorithm 2** Adaptive momentum estimation (ADAM) optimizer based on [32].

Here  $g_t^2$  denotes the element-wise square  $g_t \odot g_t$ , and all operations on vectors are element-wise.  $\beta_1^t$  and  $\beta_2^t$  denote the  $t$ -th power of  $\beta_1$  and  $\beta_2$ .

---

**Input:** Learning rate  $\nu$ , exponential decay rates for the moment estimates  $\beta_1, \beta_2 \in [0, 1)$ , scalar  $\epsilon$

**Output:** Parameters  $\theta$

Initialization: parameters  $\theta_0$ , first moment vector  $m_0 = 0$ ,  
second moment vector  $v_0 = 0$ , timestep  $t = 0$

**for**  $iter = 1, 2, 3, \dots$  **do**

Update  $t$  with  $t \leftarrow t + 1$

Compute gradient estimate:  $g_t = \nabla_{\theta} \mathcal{L}(\theta_{t-1})$

Update biased first moment estimate  $m_t$  with  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$

Update biased second raw moment estimate  $v_t$

with  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$

Compute bias-corrected first moment estimate:  $\hat{m}_t = m_t / (1 - \beta_1^t)$

Compute bias-corrected second raw moment estimate:  $\hat{v}_t = v_t / (1 - \beta_2^t)$

Update parameters  $\theta$  with  $\theta_t \leftarrow \theta_{t-1} - \nu \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$

**if**  $\theta$  converged **then**

break

**end if**

**end for**

---

## 2.2 Convolutional neural network

When the input data of neural network is an image, convolutional neural networks (CNNs) are more efficient to process it [30]. CNNs have achieved great success in the field of computer vision [3, 18], like image recognition and classification. One of the main advantages of CNNs over ANNs is that the spatial information is saved and handled by the convolution operation. Besides, the use of shared parameters in CNNs helps to save time and space [31]. As will be explained below, a deep architecture of CNNs typically consists of alternating convolutional layers and pooling layers, followed by fully connected layers. Also, a deeper network could extract more obscure information from images in general [18]. The following sections present a brief introduction of layers in CNNs. For further study, [31] is a useful resource available online.

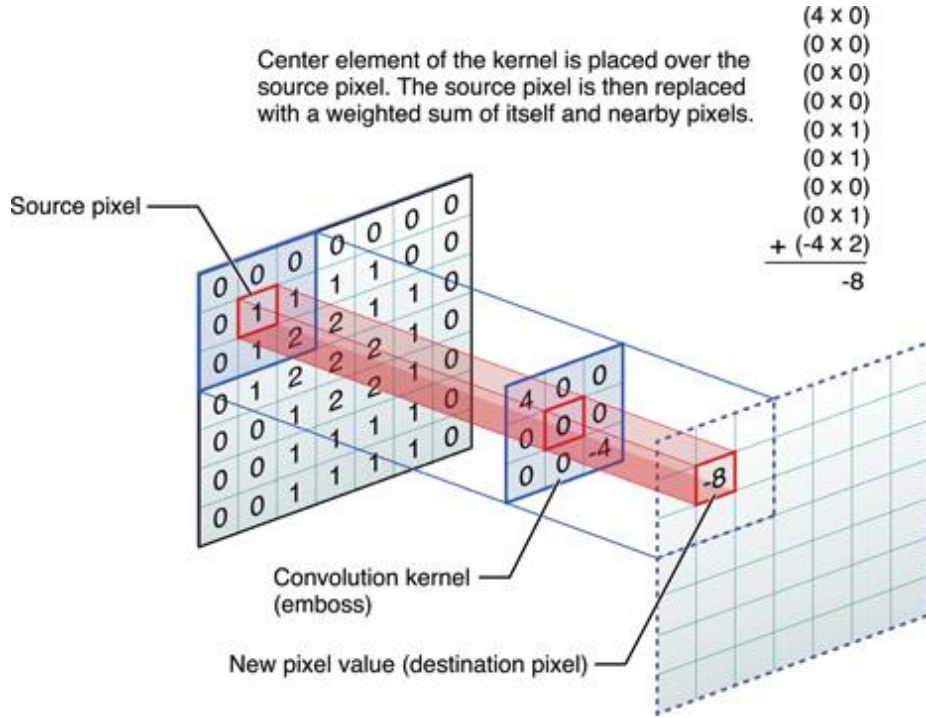
### 2.2.1 Convolutional layer

The convolution operation is the core of a CNN and is conducted in convolutional layers. It means that convolution kernels (or filters) of different sizes and strides are slid over the image and applied as dot products with pixel values of the image, as shown in Figure 2.3. The width and height of the kernel are usually equal, such as  $3 \times 3$  and  $5 \times 5$ , and the depth is equal to the number of channels of the input image. In the input layer, a grayscale image has one channel and a color image usually has three channels: red, green, and blue. As for hidden layers, the depth of input is equal to the number of kernels from last convolutional layer, since one kernel corresponds to one output channel.

Stride and padding are two standard parameters which impact the convolutional layer [31]. The stride of a kernel determines how much it moves over the image when sliding. A stride of 1 means the kernel moves pixel by pixel. The padding determines how to deal with the edge pixels in the image. When the kernel applies convolution to the edge pixels, part of the kernel is outside the image range. A common



strategy to add zeros around the image and this is called zero-padding. Also, the information of edge pixels is saved through padding.



**Figure 2.3** An illustration of convolution operator taken from [33]. A  $3 \times 3$  kernel slides over the source image and generates new pixel values to output feature maps. The new value is a weighted sum of source pixels within the current neighborhood. It saves the spatial information of source image.

The size of output feature map can be computed using the following equation, where  $h$  and  $w$  denote the height and width (usually equal) respectively, and  $padding$  denotes the number of padding pixels.

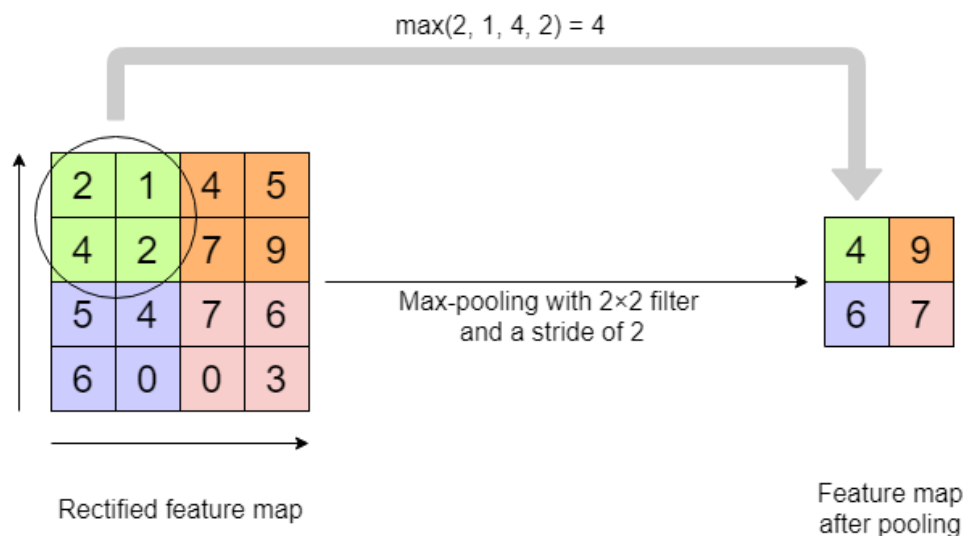
$$\begin{cases} h_{output} = \left\lfloor \frac{h_{input} - h_{kernel} + 2 \times padding}{stride} \right\rfloor + 1 \\ w_{output} = \left\lfloor \frac{w_{input} - w_{kernel} + 2 \times padding}{stride} \right\rfloor + 1 \end{cases} \quad (2.9)$$

Furthermore, a convolutional layer is usually followed by a ReLU layer,

which applies the ReLU function to the output. Because there is a general problem for both sigmoid and tanh functions, which is that they saturate [31]. This problem limits the sensitivity and removes some information, such as gradient, of an image.

### 2.2.2 Pooling layer

After the output of convolution layer is applied to an activation function, it is time to sub-sample the feature maps, which is called pooling. These original feature maps are sensitive to the position of the feature, and pooling is an effective method to reduce this sensitivity by sub-sampling [31]. There are two types of pooling method: average pooling and maximum pooling (or max-pooling), and the latter one is commonly used in CNN. Average pooling means to choose the average presence of a feature while maximum pooling means to choose the max activated presence [31]. Figure 2.4 illustrates an example of max-pooling.



**Figure 2.4** An illustration of max-pooling

A slight translation of the input image might not change the result of max-pooling, which is called “local translation invariance” [31]. Also, pooling can reduce the computational complexity during training

phase at the same time. The main parameters of a pooling layer are filter size and stride. In general, the height and width of the filter are same, and the stride is also equal to the side length of filter. The size of output feature map of pooling layer can be computed by following equation, where  $h$  and  $w$  denote the height and width (usually equal) respectively.

$$\begin{cases} h_{output} = \left\lceil \frac{h_{input} - h_{filter}}{stride} \right\rceil + 1 \\ w_{output} = \left\lceil \frac{w_{input} - w_{filter}}{stride} \right\rceil + 1 \end{cases} \quad (2.10)$$

## 2.3 Metric learning

The task of metric learning (or distance metric learning) is to learn a distance function which measures the similarity between objects [4]. The distance function can map the original data to an embedding space, where the distance is a metric of similarity. Most methods of metric learning train the model using either of following information [4]:

- Positive and negative pairs (must-link / cannot-link constraint):

$$\begin{cases} \mathcal{S} = \{(x_i, x_j) \mid x_i \text{ and } x_j \text{ should be similar}\} \\ \mathcal{D} = \{(x_i, x_j) \mid x_i \text{ and } x_j \text{ should be dissimilar}\} \end{cases} \quad (2.11)$$

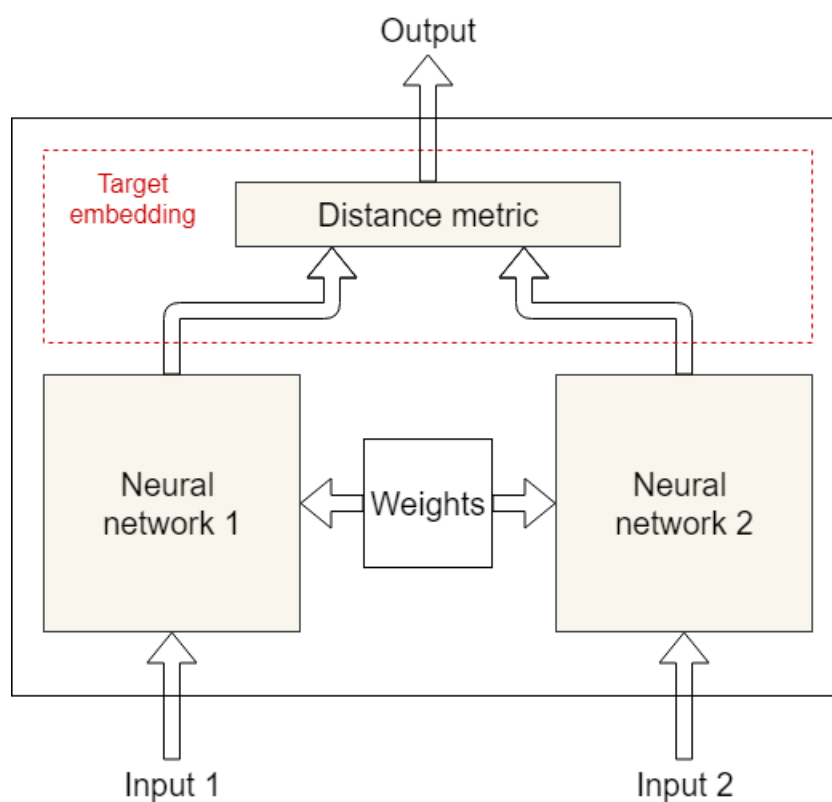
- Training triplets (relative constraint):

$$\mathcal{R} = \{(x_a, x_+, x_-) \mid x_a \text{ should be more similar to } x_+ \text{ than to } x_-\} \quad (2.12)$$

The following section presents a brief description of a Siamese architecture for metric learning and a loss function suitable for it. For further study, [4] is good material available on the internet.

### 2.3.1 Siamese network

A Siamese network is a pair of neural networks which share the same weights when applied to two different inputs to get comparable outputs [5]. It can be regarded as a distance function of metric learning: The two inputs are mapped into the output embedding by a Siamese network, and there is a loss function in the new embedding to evaluate the similarity of inputs. The common architecture is shown in Figure 2.5.



**Figure 2.5** A typical Siamese architecture based from [5]. Two networks share the same weights and map two different inputs into a comparable target embedding. The output of network is the similarity metric in target embedding. Furthermore, the metric of distance in target embedding is replaced by a loss function layer during the training phase to evaluate the network.

The neural network in Siamese architecture is not limited to ANN [11]. A CNN is usually combined with Siamese network when dealing with the image similarity task [5, 10, 12]. Then the vector in target embedding represents a metric of image similarity.

### 2.3.2 Contrastive loss

Contrastive loss function is a common loss function in Siamese networks, which can effectively deal with the relationship between pair-data, proposed by Hadsell, R. et al. in 2006 [34]. It evaluates the loss of similar and dissimilar pairs separately. In general, the loss decreases when the distance between features in a similar pair in the target embedding is small or features in a dissimilar pair is large, and vice versa. The mathematical expression is written as Equation 2.13.

$$\mathcal{L}_{contrastive} = \frac{1}{2N} \sum_{i=1}^N [y_i d_i^2 + (1 - y_i) \max(\text{margin} - d_i, 0)^2] \quad (2.13)$$

In the expression,  $d_i$  denotes the Euclidean distance between the  $i$ -th output pair feature vectors and  $\text{margin}$  denotes the threshold between similarity and dissimilarity. Besides,  $y_i = 1$  when the sample pair is similar and  $y_i = 0$  when they are dissimilar.

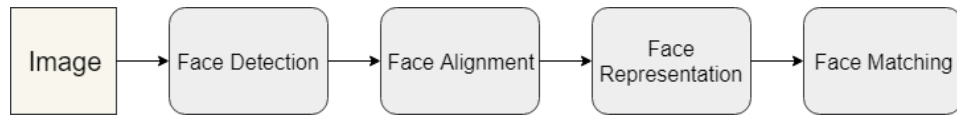
# Chapter 3

## Related work

In this chapter, a detailed introduction of related work is presented along with their relevance to this degree project. There are two main fields relevant to the thesis: First, this project aims to deal with facial images, so it is closely related to face recognition. Furthermore, some methods for image similarity might be helpful to address the face similarity task.

### 3.1 Face recognition

In the field of computer vision, face recognition is one of the most popular tasks [6]. As a well-developed technology, face recognition has been widely applied in our lives, like the security check gate in an airport and the facial lock in mobile phone. The general task of face recognition is to identify or verify the identity of one or more persons in a static image or a dynamic sequence according to a pre-stored dataset [6]. There are four main steps of this task: face detection, face alignment, face representation, and face matching, which are shown in Figure 3.1.

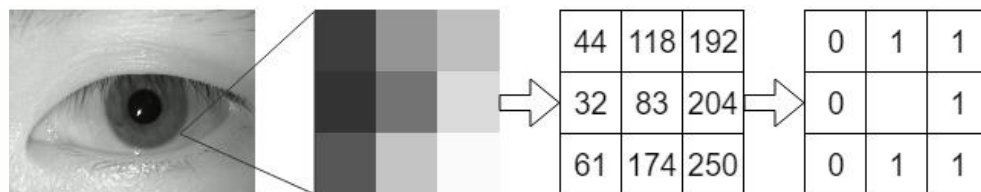


**Figure 3.1** The basic procedure of face recognition

### 3.1.1 Different methods for face recognition

At an early stage, researchers were focused on feature extraction from facial images. One of the earliest methods is to extract features from geometric parameters, which was proposed by Bledsoe in 1966 [20]. The parameters contain the distance between eyes, the height of nose, the width of head. This work is semi-automatic since these parameters are all collected under the help of manual localization, such as the location of the top of the nose.

To avoid positioning the keys points from facial images, researchers began to extract advanced features from image pixels and other domains transformed from the image until deep learning appeared. This type of features holds underlying physical characteristics rather than certain semantic information of the facial image. Common underlying characteristics include intensity, transformation coefficients (such as discrete cosine transform [21], wavelet transform [22], and Gabor transform [23]) and local texture feature (such as scale-invariant feature transform [24], histograms of oriented gradients [25] and local binary patterns [26]). Figure 3.2 is an example of how local binary patterns (LBP) works over a grayscale image.



**Figure 3.2** A typical LBP operator with a size of  $3 \times 3$ . The intensity of the center pixel is regarded as a threshold of this window. The around pixels whose intensity is less than the threshold are tagged as 0 while those greater are tagged as 1.

In addition to extracting feature from other domains, methods based on subspace analysis are also developed at the same time. Researchers attempt to reduce the dimension of features from origin image space and transform them into a new subspace, in which the features can represent the origin face more efficiently. Examples of this type of method include eigenfaces [14], Fisherfaces [15], and Laplacian faces [16]. All of these algorithms focus on saving the most important facial information in the low-dimension feature when the dimension is compressed.

The idea to apply neural network on face recognition was proposed in 1997 by Lin et al. [27]. They designed a probabilistic decision-based neural network to deal with the recognition task. Since the computational ability was limited by hardware at that time, the structure of this network was simple, and the size of dataset was small. Similarly, the following method [5] based on the neural network proposed at that period failed to make breakthrough progress in the field of computer vision.

In recent years, with the development of hardware and software, the computational ability of computers is strengthened dramatically. The structure of neural networks has become deeper and more complex and convolutional neural networks (CNNs) has gradually become applicable to computer vision. The main advantage of using CNNs is that researchers do not need to design features manually anymore since the network can learn a specific feature for each task based on the image dataset by itself.

Some networks achieve acceptable results in face recognition. DeepFace proposed by a Facebook group [18] reaches an accuracy of 97.35% on the Labeled Faces in the Wild (LFW) [29] dataset. The input images of this network need to be aligned in advance. FaceNet proposed by Schroff et al. [28] reaches a higher accuracy of 99.63% on LFW. And FaceNet is trained on unaligned images with triplet loss. VGG-Face proposed by Parkhi et al. [3] achieves an accuracy of 99.13% on LFW, which is trained on a smaller dataset but get similar accuracy with other networks.



### 3.1.2 VGG-Face

VGG-Face network is a “very deep” convolutional neural network which achieves similar accuracy with smaller dataset [3]. Parkhi et al. proposed three similar architectures of VGG-Face, in which the 16-layer one is used to extract VGG-Face CNN descriptors. Figure 3.3 shows the specific configuration of the 16-layer VGG-Face network.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	-	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	-	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	-	3	-	64	-	-	64	-	128	-	-	128	-	256	-	256	-	-	256
num filts	-	64	-	64	-	-	128	-	128	-	-	256	-	256	-	256	-	-	512
stride	-	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	-	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1
layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmax
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	conv	relu7	conv
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	-	512	-	512	-	-	512	-	512	-	512	-	-	512	-	4096	-	4096	-
num filts	-	512	-	512	-	-	512	-	512	-	512	-	-	4096	-	4096	-	2622	-
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

**Figure 3.3** Network configuration of 16-layer VGG-Face taken from [3].

There are 16 convolutional layers in total, and all of them are followed by a ReLU layer, respectively. The last three layers are fully connected (FC) layers that are annotated as “conv” in the configuration since the filter matches the size of input data [3]. Furthermore, the input to this network is a  $224 \times 224$  facial image with the average image subtracted [3]. Also, the output of the last FC layer is a vector of 2,622 elements, which can be regarded as an embedding space in metric learning.

### 3.1.3 The difference between face recognition and similarity

Although face similarity is relevant to face recognition, a network explicitly trained for recognition might be improper for a similarity task. This intuition was proven by Sadvnik et al. in 2018 [1]. They conducted an experiment to measure the difference between recognition and similarity by collecting a novel dataset.

First, they use the VGG-Face descriptor to process the facial images in the dataset and map them into the embedding space mentioned in Section 3.1.2. Then they compute the Euclidean distance of feature

vectors for each pair of images and bin them into 10 different bins according to their distance. To compare pairs from different bins, they select 100 test cases from each bin for comparison. For example, if the face recognition algorithm were a good measurement of face similarity, the image pair of small-distance bin would look more similar than that of large-distance bin. Therefore, they have a total of  $100 \times (10|2) = 4500$  test cases. The comparison results are shown as a matrix in Figure 3.4.

	0.95	1.0	1.05	1.1	1.15	1.2	1.25	1.3	1.35	1.4
0.95		26	28	35	38	49	73	71	81	94
1.0	17		22	26	34	47	60	56	82	86
1.05	13	16		22	21	40	58	56	76	87
1.1	9	18	17		27	33	46	55	68	83
1.15	7	13	17	14		30	48	49	62	70
1.2	10	12	6	12	19		40	48	62	67
1.25	2	7	5	6	14	13		28	42	56
1.3	5	3	4	7	4	14	27		41	60
1.35	1	1	0	2	7	9	12	12		39
1.4	0	0	1	3	2	2	9	11	21	

**Figure 3.4** The result of comparison experiment taken from [1]. The number shows the frequency that the row bins are chosen as more similar over the column bins. Besides, the results with less than 80% agreement are ignored.

There is a strong correlation between similarity and recognition from the comparison between small-distance bin and large-distance bin, which is shown as the upper right corner of the matrix. However, when it comes to the comparison between bins with small distance, the task for face recognition cannot reflect the similarity accurately, which is shown as the upper left corner of the matrix. In conclusion, the recognition embedding can separate the most dissimilar ones from somewhat similar ones, but it does not work well at finding the most similar images [1].

## 3.2 Image similarity

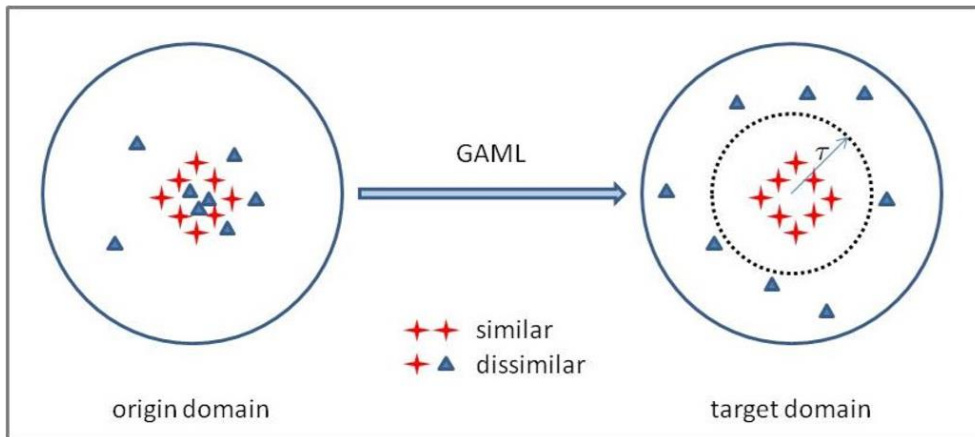
Measuring similarity between two images is another important task in the field of computer vision [7, 8]. It has played an important role in object classification and image retrieval. In general, image similarity contains two types: semantic similarity and visual similarity. The former means the classification relationship between images, like which super-class the two images belong to. The latter one means subjective feelings of similarity on an image pair. [9] proposed that there two types of similarity are both critical by evaluating the impact of each.

At an early stage, researchers choose features extracted from images to compare similarity, such as texture [35] and scale-invariant feature transform (SIFT) [24]. To obtain a higher level of semantic concepts, the outputs of a classifier [7] are used as features, since they contain the classification information of an image. These methods concentrate more on the semantic similarity but ignore the visual similarity.

In recent years, image similarity is developed based on metric learning. As introduced in Section 2.3, the purpose of metric learning is to find a metric or embedding space to measure the similarity. Cosine distance is one of the metrics measuring the relationship between features, which is defined as Equation 3.1.

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} \quad (3.1)$$

Zhang, N. et al. proposed geometry-aware metric learning (GAML) based on cosine distance in 2016 [2]. They use a fully connected network to transform the features extracted from images into a new embedding, in which the distance between features in a dissimilar pair is enlarged while the distance between features in a similar pair is preserved not narrowed, as shown in Figure 3.5. More details of the implementation of GAML is discussed in Section 4.2.



**Figure 3.5** An illustration of GAML taken from [2]. In the origin domain, similar and dissimilar samples are located near each other. After the transformation of GAML, the distance between similar samples is preserved but dissimilar samples are pulled away with a threshold  $\tau$ .

Since this project focuses on face similarity, methods for image similarity might be somewhat useful to explore measuring metric. However, they are two distinct tasks because human-specific neural processing for faces differs from other objects [6]. Besides, measuring the face similarity is somewhat more subjective than other recognition tasks and this project is aimed to explore the subjective bias.

# Chapter 4

## Methods

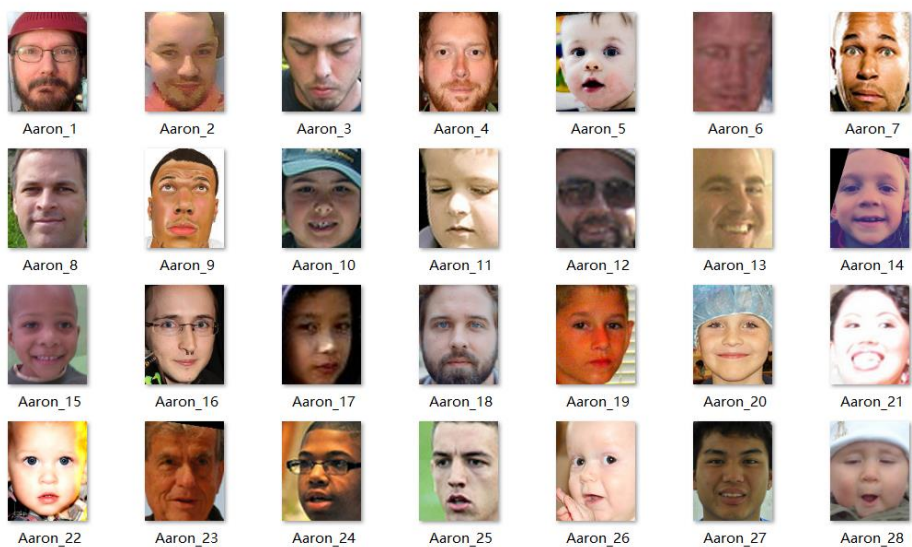
This chapter presents a detailed introduction about how this degree project is implemented. The project can be divided into two main parts: data collection and network training. Two different Siamese networks are used to compare the measurement results, which are implemented with CNN and metric learning, respectively. These two methods are presented in separate sections later. Also, the evaluation methods about the experiment results are provided in this chapter.

### 4.1 Data collection

Machine learning is dependent on plenty of data, so plenty of ground-truths of face similarity is needed before network training. Unfortunately, there is no public dataset which contains information about similarity. A decision was thus made to collect a novel dataset that is specific for this task.

Since this novel dataset should include information on human judgement about whether two facial images look similar, there are two indispensable elements which are needed to prepare in advance. One of them is a proper raw dataset that contains plenty of facial images from different identifies, and the other one is a platform which provides an efficient way to collect human judgements.

The Names 100 dataset [19] is chosen as the raw dataset for the following reasons. First, this dataset is organized into 100 different names. Each name contains 800 different facial images from social networks. So there are 80,000 facial images altogether contained in this dataset, which is larger than some common datasets. Second, all the facial images in this dataset are aligned and resized into  $120 \times 150$  pixels, and thus I could save some time from face detection and alignment and focus more on face similarity. Third, since the images in the dataset are collected from social networks, there are no image of celebrities included in it. People might make different judgements on similarity when it involves identities they know about. So a wild dataset can prevent the impact of this type of deviation. Some facial images of this database are shown in Figure 4.1. As for the platform for this task, I choose to ask workers on Amazon Mechanical Turk to help annotate these images.

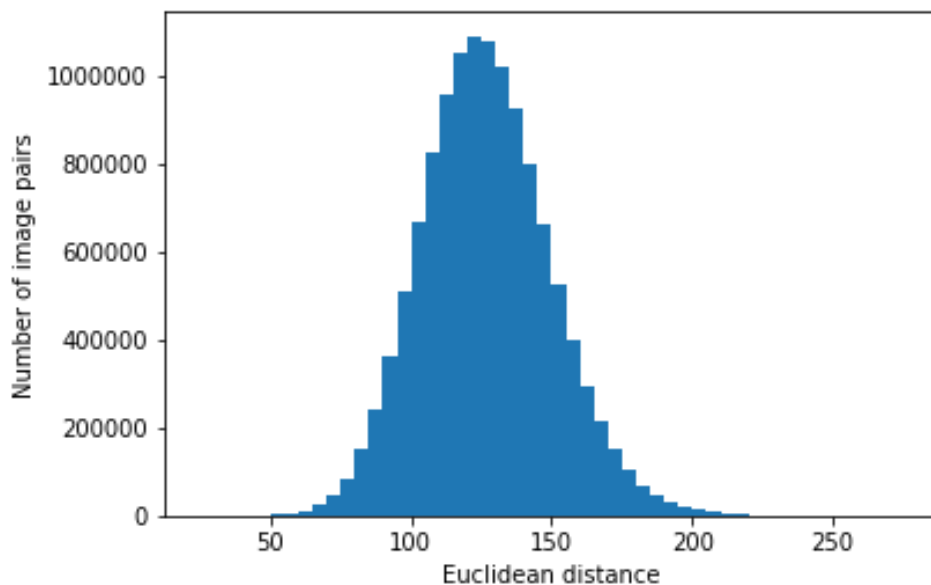


**Figure 4.1** A list of first 28 images in Names 100 dataset. These images are collected from different identities who have the same name “Aaron”. All of the identities are distinguished by name in this dataset.

The first step is to map these images into an embedding space for the following processing. I randomly select 5,000 facial images from the

dataset and then use the pre-trained VGG-Face CNN descriptor [3] to extract features from images. The meaningful embedding space in this project is the embedding before the last softmax layer, so I remove the last layer and use the output of the penultimate layer as the final feature. After the VGG-Face descriptor has processed all facial images, I get 5,000 feature vectors with 2,622 elements from the output.

The next step is to screen similar potential pairs from the embedding space. As introduced in Section 3.1.3, if the features extracted by the VGG-Face descriptor from two images have a small Euclidean distance, these two images are more likely to look similar. In contrast, the image pair whose feature distance is significant from each other have little possibility to look similar. So I decide to choose the pairs with the smallest distance as the potential pairs. Since identities are only distinguished by names in the dataset, I calculate the pair distance between features whose names are different and thus it is ensured that the two images are not from the same person. I finally get 12,372,666 Euclidean distances from different feature pairs and Figure 4.2 shows the distribution of the pair-distances through a histogram.



**Figure 4.2** Distribution of pair-distance of 5000 VGG features

Generally, more ground-truths are beneficial to train a neural network, but the cost of human annotation increases simultaneously. To make a tradeoff between the number of potential pairs and the cost according to the budget from my principal, I choose a pair-distance of 56 as the threshold to screen the pairs. Any image pair whose distance is less than or equal to the threshold is collected for the following annotation and the total number is 5,238. Before publishing the annotation task on Mechanical Turk, all the similar potential pairs should be uploaded and stored on the internet with public access. I choose Amazon S3 as the cloud storage and get a list of URLs from the storage bucket.

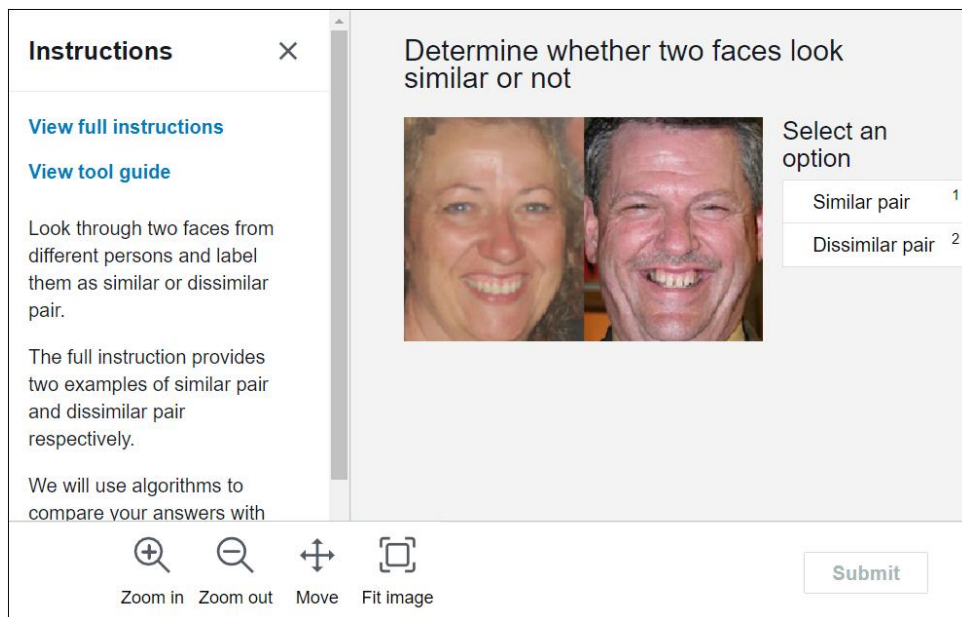
The following step is to design a task on Mechanical Turk to collect ground-truths of similar and dissimilar facial image pairs, and the task should satisfy the following conditions:

- The whole task should be divided into independent Hits, and one Hit corresponds to one image pair.
- Each Hit should collect information about “similar pair” or “dissimilar pair”.
- Since the annotation of similarity is subjective, each Hit should be annotated multiple times from different workers to reduce the impact of subjective bias.
- It could be better to provide examples of similar pairs and dissimilar pairs to help workers have an intuitive understanding of this task.
- The choices which are annotated within less than 1 second might be rejected and removed to avoid quick clicks without consideration.

Therefore, I decide to frame this task as a labeling task. It consists of a brief description and a pair of facial images. As shown in Figure 4.3, there are two options to choose for each pair: “similar pair” and “dissimilar pair”. Also, a pair of examples (a similar pair and a



dissimilar pair) are displayed in full instructions for this task. Since there are 5,238 image pairs to be annotated, I divide them and get a total of 5,238 Hits. Considering the least reward per assignment per worker is \$0.01, each Hit is distributed to 3 different workers to save costs. Since it is not allowed to upload more than 500 Hits in one batch, I randomly divide the 5,238 image pairs into 11 different batches and publish them separately.



**Figure 4.3** An example Hit of Mechanical Turk task

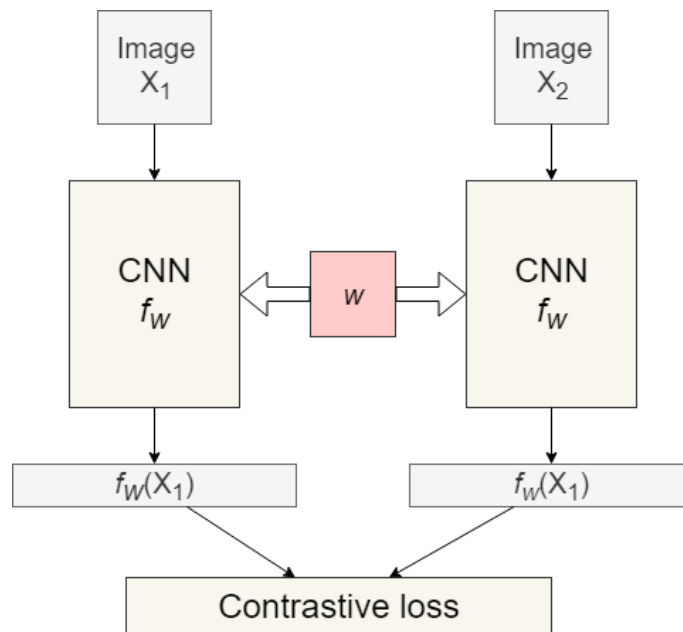
After all of Hits are finished by workers, 15,714 results are collected into 11 different CSV files from Mechanical Turk, which consist of annotations of similar pair and dissimilar pair. For each image pair, if it is annotated as a similar pair two or three times, then this pair is labeled as a similar pair in the dataset. In contrast, if an image pair is annotated as a dissimilar pair two or three times, then it gets a label of dissimilar pair in the dataset. Finally, 1,750 similar image pairs and 3488 dissimilar image pairs are collected in the novel dataset.

## 4.2 Experiment approaches

Once the novel dataset for face similarity is collected, two Siamese networks with different methods are built: convolutional neural network (CNN) and geometry-aware metric learning (GAML). The former one processes input facial images directly and outputs features used for similarity comparison. The latter network uses VGG-Face features as input and maps them into a new embedding space in which the distance represents similarity between images.

### 4.2.1 Experiment 1: Measuring similarity based on CNN

As discussed in Section 2.2 and 2.3, CNN can save and extract spatial information from original images, and a Siamese network is good at finding similarity between two comparable things. So this method combines these two networks to find a function which can measure the similarity. The architecture of this learning machine is shown in Figure 4.4.



**Figure 4.4** The architecture of Siamese CNN

At the top of this network,  $X_1$  and  $X_2$  denote a pair of facial images shown as an input to the network. Moreover there is a corresponding label  $y$  for each image pair,  $y = 1$  if  $X_1$  and  $X_2$  are similar pair and  $y = 0$  otherwise. Next,  $f_w$  denotes the mapping function which represents the CNN and  $w$  denotes the shared parameter of Siamese network. After that,  $f_w(X_1)$  and  $f_w(X_2)$  are two features that are generated by mapping  $X_1$  and  $X_2$ . At last, Euclidean distance of  $f_w(X_1)$  and  $f_w(X_2)$  is used to evaluate the contrastive loss. The contrastive loss in this experiment is defined as:

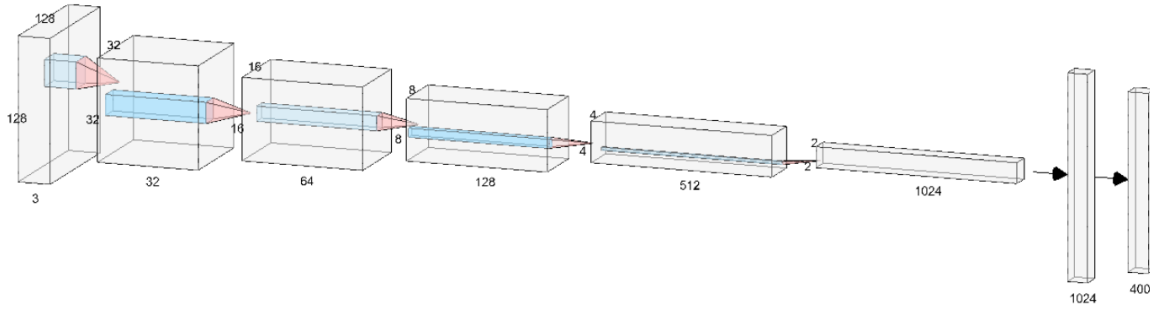
$$J = \frac{1}{2N} \sum_{i=1}^N [y_i d(X_1, X_2, f_w)_i^2 + (1 - y_i) \max(\tau - d(X_1, X_2, f_w)_i, 0)^2] \quad (4.1)$$

where  $(X_1, X_2, f_w)_i$  denotes the  $i$ -th sample and  $d(X_1, X_2, f_w)$  is the Euclidean distance between feature vectors  $f_w(X_1)$  and  $f_w(X_2)$ , which is computed as:

$$d(X_1, X_2, f_w) = \|f_w(X_1) - f_w(X_2)\| \quad (4.2)$$

The loss of similar pairs ( $y_i = 1$ ) is evaluated by  $d(X_1, X_2, f_w)^2$ , which means the distance between similar pairs should be narrowed after being mapped by the network. The loss of dissimilar pairs ( $y_i = 0$ ) is evaluated by  $\max(\tau - d(X_1, X_2, f_w), 0)^2$ , which means the distance between dissimilar pairs should be enlarged over a margin  $\tau$  after being processed by CNN.

The architecture of the CNN consists of 6 convolutional layers and one fully connected layer. Each convolutional layer is followed by a ReLU layer for activation and a max-pooling layer for sub-sampling. A simple illustration of this architecture is shown in Figure 4.5. In the following detailed description,  $C_x$  represents convolutional layer,  $P_x$  represents max-pooling layer, and  $F_x$  represents fully connected layer, where  $x$  is the layer index.



**Figure 4.5** A simple illustration for CNN architecture. There are in total 6 convolutional layers and one fully connected layer. The penultimate layer is a flatten layer, and pooling layers and ReLU layers are omitted. The creation tool for this figure is available in [36].

The whole architecture of CNN is  $C_1(P_1) - C_2(P_2) - C_3(P_3) - C_4(P_4) - C_5(P_5) - C_6(P_6) - F_7$ :

- $C_1$ , Feature maps: 32; Size:  $64 \times 64$ ; Kernel size:  $9 \times 9$ ; Stride: 2; Padding strategy: 'same'; Activation function: ReLU.  
Fully connected with the input.  
 $P_1$ , Feature maps: 32; Size:  $32 \times 32$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: 'same'.
- $C_2$ , Feature maps: 64; Size:  $32 \times 32$ ; Kernel size:  $7 \times 7$ ; Stride: 1; Padding strategy: 'same'; Activation function: ReLU.  
 $P_2$ , Feature maps: 64; Size:  $16 \times 16$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: 'same'.
- $C_3$ , Feature maps: 128; Size:  $16 \times 16$ ; Kernel size:  $5 \times 5$ ; Stride: 1; Padding strategy: 'same'; Activation function: ReLU.  
 $P_3$ , Feature maps: 128; Size:  $8 \times 8$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: 'same'.
- $C_4$ , Feature maps: 256; Size:  $8 \times 8$ ; Kernel size:  $3 \times 3$ ; Stride: 1; Padding strategy: 'same'; Activation function: ReLU.  
 $P_4$ , Feature maps: 256; Size:  $4 \times 4$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: 'same'.
- $C_5$ , Feature maps: 512; Size:  $4 \times 4$ ; Kernel size:  $1 \times 1$ ; Stride: 1;

Padding strategy: ‘same’; Activation function: ReLU.

$P_5$ , Feature maps: 512; Size:  $2 \times 2$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: ‘same’.

- $C_6$ , Feature maps: 1,024; Size:  $2 \times 2$ ; Kernel size:  $1 \times 1$ ; Stride: 1; Padding strategy: ‘same’; Activation function: ReLU.

$P_6$ , Feature maps: 1,024; Size:  $1 \times 1$ ; Field of view:  $2 \times 2$ ; Stride: 2; Padding strategy: ‘same’.

Followed by a flatten layer and then fully connected to  $F_7$ .

- $F_7$ , Number of units: 400; Connections: 410,000.

The corresponding parameter setting and training protocol will be described in Section 4.3. After training, the output vector of the CNN can be regarded as a feature of an image. Moreover, the Euclidean distance between these features represents the similarity between facial images.

## 4.2.2 Experiment 2: Measuring similarity based on GAML

At the stage of data collection, I have already extracted VGG-Face features from facial images. As discussed in Section 3.1.3, the distance between VGG-Face features is not a representation of similarity, so I apply GAML to map these features into a new embedding space in this experiment. I make some adjustment on the original loss function in [2] to maintain consistency with pre-processing in data collection, which is formulated as the following equations:

$$J = J_1 + \alpha J_2 \quad (4.3)$$

$$J_1 = \frac{1}{2} \sum_i S_i [d(\mathcal{N}(x_i), \mathcal{N}(y_i), f_{w,b}) - d(\mathcal{N}(x_i), \mathcal{N}(y_i))]^2 \quad (4.4)$$

$$J_2 = \sum_i D_i \max[\tau - d(\mathcal{N}(x_i), \mathcal{N}(y_i), f_{w,b}), 0] \quad (4.5)$$

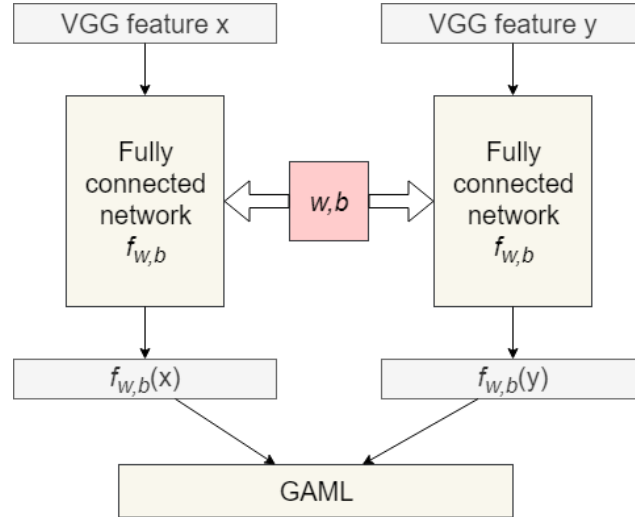
In these equations,  $d(\mathcal{N}(x_i), \mathcal{N}(y_i))$  and  $d(\mathcal{N}(x_i), \mathcal{N}(y_i), f_{w,b})$

denote the Euclidean distance of two normalized vectors in the original embedding and target embedding respectively, which is defined as follows:

$$d(\mathcal{N}(x_i), \mathcal{N}(y_i)) = \left\| \frac{x_i}{\|x_i\|} - \frac{y_i}{\|y_i\|} \right\| \quad (4.6)$$

$$d(\mathcal{N}(x_i), \mathcal{N}(y_i), f_{w,b}) = \left\| \frac{f_{w,b}(x_i)}{\|f_{w,b}(x_i)\|} - \frac{f_{w,b}(y_i)}{\|f_{w,b}(y_i)\|} \right\| \quad (4.7)$$

The network is regarded as a function  $f_{w,b}$ , in which weight  $w$  and bias  $b$  are parameters of the network.  $J_1$  is the loss evaluation for similar pairs, which maintains distances between similar pairs instead of narrowing. Therefore, the loss increases if the geometrical shape (distance in origin domain) of similarity is changed.  $S_i$  in Equation 4.4 is set as one if  $(x_i, y_i)$  is a similar pair, otherwise it is set as zero.  $J_2$  is the loss evaluation for dissimilar pairs, which tries to enlarge the distance between dissimilar pairs over a threshold  $\tau$ .  $D_i$  in Equation 4.5 is set as one if  $(x_i, y_i)$  is a dissimilar pair, otherwise it is set as zero. The architecture of this learning machine is shown in Figure 4.6.



**Figure 4.6** The architecture of Siamese GAML

The neural network has four fully connected layers, which maps the VGG feature into a new embedding. And each layer is followed by a non-linear activation function. The detailed information for these layers  $F_1 - F_2 - F_3 - F_4$  are listed as follows:

- $F_1$ , Number of units: 2,000; Activation function: tanh  
Connections: 5,246,000.  
Fully connected with the input.
- $F_2$ , Number of units: 1,000; Activation function: tanh  
Connections: 2,001,000.
- $F_3$ , Number of units: 600; Activation function: tanh  
Connections: 600,600.
- $F_4$ , Number of units: 400; Activation function: tanh  
Connections: 240,400.

The corresponding parameter setting and training protocol are described in the next section. After training, the output embedding can be regarded as a space for similarity measuring. Moreover, the Euclidean distance between the output vectors represents the similarity between facial images.

## 4.3 Experiment setting

This experiment is carried on 8 vCPUs with 25GB of RAM and an NVIDIA Tesla K80 GPU with 6GB of onboard memory. Moreover, the operation system is Ubuntu 16.04 LTS. The implementation is based on Python 3.5 and TensorFlow docker [37]. Following is the detailed description of implementation.

### 4.3.1 Dataset segmentation

Since there is no other available dataset for face similarity, I test both networks on my own novel similar face dataset collected from Names 100 Dataset. According to the input of two networks, image pairs should be pre-processed and partitioned correspondingly.

In experiment 1, the input of the Siamese CNN is a facial image pair. Before partitioning, each image should be rescaled into a size of  $128 \times 128$  in RGB. Then the whole dataset is randomly divided into three sets with a ratio of 8:1:1, which are training set, validation set, and test set respectively. The training set contains 4190 image pairs with their labels (1 means similar and 0 means dissimilar) and the other two sets both contain 524 pairs.

In experiment 2, the input of Siamese fully connected network is VGG feature pair. First, I apply VGG-Face descriptor to each image and extract feature vectors with 2,622 elements. Then I segment the feature pairs into three sets with a ratio of 8:1:1, which have the same size as experiment 1. Furthermore, there are two labels  $S_i$  and  $D_i$  for each feature pair used in the calculation of GAML loss. So I create these labels correspondingly and attach them with feature pairs.

### 4.3.2 Training parameters and protocol

Table 2 is a list of detailed experiment configuration, including necessary parameters and training protocols.

**Table 2** Experiment configuration

	<b>Siamese CNN</b>	<b>Siamese GAML</b>
Activation function	ReLU	tanh
Parameter initialization	Weight $w$ is initialized with Xavier initializer [38]	Weight $w$ is initialized with uniform distribution; Bias $b$ is initialized with 0
Optimizer	Momentum	ADAM
Learning rate	0.01	0.0001
Batch size	500	100
Epoch	400	200
Other parameters	Margin $\tau$ is set to 2.0	Balancing parameter $\alpha$ is set to 0.5; Threshold $\tau$ is set to 0.55



## 4.4 Evaluation metrics

The test result of each network model is evaluated according to the following metrics:

A confusion matrix is a common metric in machine learning, in which each index is defined according to the relationship between prediction result and ground-truth. Table 3 shows the specific definition of each index.

**Table 3** Confusion matrix

<b>Ground-truth Prediction</b>	<b>Similar</b>	<b>Dissimilar</b>
<b>Similar</b>	True positive (TP)	False positive (FP)
<b>Dissimilar</b>	False negative (FN)	True negative (TN)

The first evaluation method is the receiver operating characteristic (ROC) curve. The curve shows the relationship between true positive rate (TPR) and false positive rate (FPR). TPR and FPR are calculated by the following equations:

$$TPR = \frac{TP}{TP + FN} \quad (4.8)$$

$$FPR = \frac{FP}{FP + TN} \quad (4.9)$$

The area under the curve of ROC (AUC-ROC) is the main evaluation metric of prediction accuracy for networks in this project. It has a range from 0 to 1. And if AUC is equal to 0.5, the model is regarded as a random classifier. If AUC is close to 1, then the model is better than random classifier. Otherwise, the model is worse if AUC is close to 0.

Another evaluation metric is Matthews correlation coefficient (MCC) [39], which is defined as Equation 4.8.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (4.10)$$

It is used to evaluate the model with a specific prediction threshold. It has a range from -1 to 1. If MCC is equal to 0, the model is regarded as a random classifier under the current threshold. If MCC is close to 1, then the model is better than random classifier under the current threshold. Otherwise, the model is worse if MCC is close to -1.

# Chapter 5

## Result and analysis

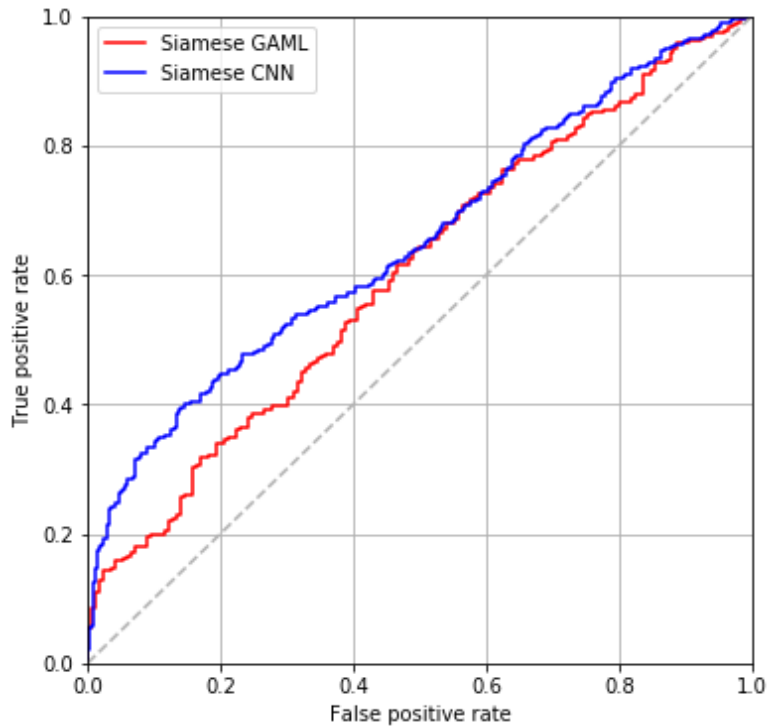
This chapter presents the results of each experiment and makes a comparison between them, including two sections. The first part shows the prediction accuracy on the novel dataset, and the second part shows the measurement results when applying the models on Names 100 Dataset.

### 5.1 Prediction result and analysis

I compare the results of two proposed models: Siamese CNN and Siamese GAML, based on the novel similarity dataset. The ROC curves are shown in Figure 5.1, and AUC-ROC accuracy of each model is presented in Table 4. The figure shows that the curve of CNN is almost above the curve of GAML over the whole range. As a result, the Siamese CNN model has higher AUC-ROC accuracy than the GAML model.

**Table 4** AUC-ROC accuracy of two face similarity models

Model	Siamese CNN	Siamese GAML
AUC-ROC	65.11%	60.20%



**Figure 5.1** ROC curves of two Siamese models on similar face dataset

With a proper classification threshold chosen according to the ROC curves, I get the MCC of each model, which is presented in Table 5. It shows that Siamese CNN model makes a better prediction on face similarity with my novel dataset.

**Table 5** MCC of two face similarity models

Model	Siamese CNN	Siamese GAML
Threshold	1.5768	0.6963
MCC	0.2547	0.1844


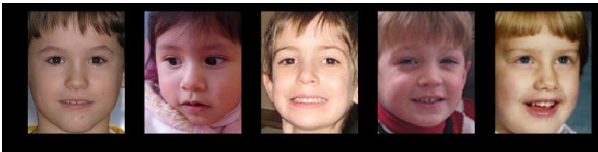



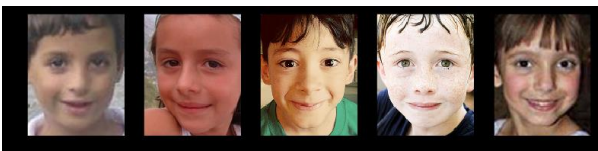
The reason why the CNN model outperforms the GAML model is complex. The most likely reason is that the novel dataset was collected from human observation so that the similarity they annotated is a kind of visual similarity. However, the input of the CNN model is a facial

image pair while that of GAML model is VGG feature pair. So the CNN model can extract more spatial information about visual similarity from image input, and as a result, it gets a higher accuracy on the face similar dataset. As for the GAML model, since it holds the distance between similar pairs, a small threshold is hard to screen the similar pair in target embedding if its distance is large in the original embedding. As a result, there is not an appropriate global threshold to screen all the similar pairs in the test set.












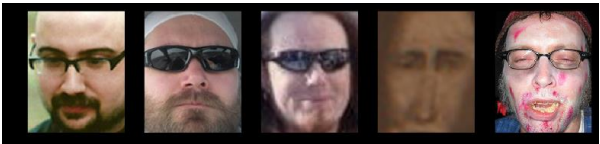
## 5.2 Similarity ranking result

In order to evaluate how the proposed models work on the original Names 100 Dataset, I compare the top-5 most similar faces (similarity ranking results) generated by the two face similarity models and VGG-Face descriptor, given a query facial image. There are some example results listed in Table 6.










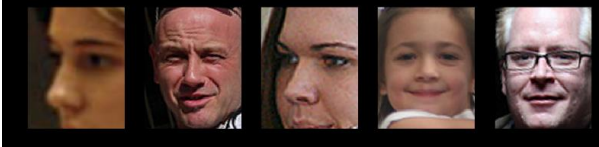


**Table 6** Examples of similarity ranking results

	Model	Query	Top-5 similar
1	VGG Face		
	Siamese CNN		
	Siamese GAML		

**Table 6 (continued)**

	Model	Query	Top-5 similar
2	VGG Face		
	Siamese CNN		
	Siamese GAML		
3	VGG Face		
	Siamese CNN		
	Siamese GAML		

**Table 6 (continued)**

	Model	Query	Top-5 similar
4	VGG Face		
	Siamese CNN		
	Siamese GAML		
5	VGG Face		
	Siamese CNN		
	Siamese GAML		

Interestingly, the Siamese CNN model seems to generate the worst results among the three methods, even though it gets a higher accuracy in the AUC-ROC evaluation. It is hard to find similarity between the query and result images. Even the gender of result images is not consistent with the query one. The results of the Siamese GAML model is sometimes an adjustment and re-ordering of VGG results. It saves the most similar ones which it “thinks” and throws away those dissimilar results, even though it has a poor performance in the AUC-ROC evaluation.

There are multiple reasons why these models have an opposite performance with ROC evaluation. First, the CNN model applies image pairs as input, so it focuses more on visual similarity and learns less about semantic similarity of faces, like gender and glasses. Second, as a face recognition descriptor, VGG-Face holds rich information about semantic similarity. The GAML model uses this feature as input and learns itself based on a dataset with annotations about visual similarity. So this model deals with both similarities at the same time. Third, as discussed in Section 3.1.3, VGG-Face has a poor performance in distinguishing similar faces with small distance. But the GAML model maintains the VGG distance between similar pairs and pushes away those dissimilar pairs. So it gets better performance than the VGG-Face descriptor.



# Chapter 6

## Conclusion

In this project, I collect a novel dataset containing image pairs with similarity information. Moreover, I construct two Siamese networks for the face similarity task, in which one is based on a convolution neural network (CNN) and the other is combined with geometry-aware metric learning (GAML). The experiment results show that the specific similarity model has a better performance than conventional face recognition methods when dealing with face similarity. Besides, the combination of metric learning with VGG-Face outperforms the independent CNN model in face similarity task.

# Chapter 7

## Future work

This project raises many questions worth investigating, and thus I believe this topic can be explored and improved further.

- The information about similarity is collected as human annotation. There might be other ways to obtain this. For example, it would be better to combine some personal data to increase the accuracy, like gender and age.
- The face similarity dataset is collected in pairs. Perhaps it would be better to collect similarity information in triplets. Since a triplet provides contrastive images with an anchor image, it will help to train a similarity ranking model.
- The CNN architecture used in experiment 1 is simple. It is beyond my ability to design a sophisticated and remarkable neural network by myself at present. A deeper network with complex architecture might have a better performance on face similarity.
- The GAML model does not get high accuracy in the ROC evaluation, even though it generates acceptable results in similarity ranking task. Is it possible to find a proper metric to evaluate this type of scaling problems?

# Bibliography

- [1] Sadvnik, A., Gharbi, W., Vu, T., & Gallagher, A. (2018). Finding your Lookalike: Measuring Face Similarity Rather than Face Identity. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. doi:10.1109/cvprw.2018.00311
- [2] Zhang, N., Han, J., Hu, J., & Deng, W. (2016). Geometry-aware metric learning for similar face recognition. *2016 IEEE International Conference on Multimedia and Expo (ICME)*. doi:10.1109/icme.2016.7552916
- [3] Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition. *Proceedings of the British Machine Vision Conference 2015*. doi:10.5244/c.29.41
- [4] Bellet, A., Habrard, A., & Sebban, M. (2015). *Metric learning*. S.l.: Morgan & Claypool.
- [5] Chopra, S., Hadsell, R., & Lecun, Y. (n.d.). Learning a Similarity Metric Discriminatively, with Application to Face Verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*. doi:10.1109/cvpr.2005.202
- [6] Sinha, P., Balas, B., Ostrovsky, Y., & Russell, R. (2006). Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About. *Proceedings of the IEEE, 94(11)*, 1948-1962. doi:10.1109/jproc.2006.884093
- [7] Rodriguez, E., Nikolaidis, K., Mu, T., Ralph, J. F., & Goulermas, J. Y. (2011). Towards collaborative feature extraction for face recognition. *Natural Computing, 11(3)*, 395-404. doi:10.1007/s11047-011-9285-6
- [8] Wang, G., Hoiem, D., & Forsyth, D. (2009). Learning image

- similarity from Flickr groups using Stochastic Intersection Kernel MACHines. *2009 IEEE 12th International Conference on Computer Vision*. doi:10.1109/iccv.2009.5459167
- [9] Deselaers, T., & Ferrari, V. (2011). Visual and semantic similarity in ImageNet. *Cvpr 2011*. doi:10.1109/cvpr.2011.5995474
- [10] Qiu, F., Kamata, S., & Ma, L. (2017). Deep Face Recognition under Eyeglass and Scale Variation Using Extended Siamese Network. *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*. doi:10.1109/acpr.2017.48
- [11] Zhao, F., Xu, J., & Lin, Y. (2018). Similarity Measure for Patients via A Siamese CNN Network. *Algorithms and Architectures for Parallel Processing Lecture Notes in Computer Science*, 319-328. doi:10.1007/978-3-030-05054-2\_25
- [12] Zhang, T., Wang, H., & Dong, Q. (2018). Deep Disentangling Siamese Network for Frontal Face Synthesis Under Neutral Illumination. *IEEE Signal Processing Letters*, 25(9), 1344-1348. doi:10.1109/lsp.2018. 2858558
- [13] Xiong, X., & Torre, F. D. (2013). Supervised Descent Method and Its Applications to Face Alignment. *2013 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2013.75
- [14] Turk, M., & Pentland, A. (n.d.). Face recognition using eigenfaces. *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.1991. 139758
- [15] Belhumeur, P., Hespanha, J., & Kriegman, D. (1997). Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), 711-720. doi:10.1109/34.598228
- [16] He, X., Yan, S., Hu, Y., Niyogi, P., & Zhang, H. (2005). Face recognition using Laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3), 328-340. doi:10.1109/tpami.2005.55
- [17] Frome, A., Singer, Y., Sha, F., & Malik, J. (2007). Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classification. *2007 IEEE 11th*

*International Conference on Computer Vision*.  
doi:10.1109/iccv.2007.4408839

- [18] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2014.220
- [19] Chen, H., Gallagher, A. C., & Girod, B. (2013). What's in a Name? First Names as Facial Attributes. *2013 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2013.432
- [20] Bledsoe, W. W. (1964). The Model Method in Facial Recognition. *Technical Report PRI 15, Panoramic Research, Inc.*, Palo Alto, California.
- [21] Baskurt, A. (1990). Numerical image compression using the discrete cosine transform. *Signal Processing*,19(4), 346. doi:10.1016/0165-1684(90)90166-v
- [22] Tjahyadi, R., & Liu, W. (n.d.). Image classification for quality compression with wavelet filters based on image feature analysis. *6th International Conference on Signal Processing, 2002*. doi:10.1109/icosp.2002.1181207
- [23] Cojoc, D., Grattoni, P., Nerino, R., & Pettiti, G. (1998). Image description using Gabor wavelets. *OPTIKA 98: 5th Congress on Modern Optics*. doi:10.1117/12.324549
- [24] Ke, Y., & Sukthankar, R. (n.d.). PCA-SIFT: A more distinctive representation for local image descriptors. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*.doi:10.1109/cvpr.2004.1315206
- [25] Déniz, O., Bueno, G., Salido, J., & Torre, F. D. (2011). Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*,32(12), 1598-1603. doi:10.1016/j.patrec.2011.01.004
- [26] Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,28(12), 2037-2041. doi:10.1109/

tpami.2006.244

- [27] Lin, S., Kung, S., & Lin, L. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1), 114-132. doi:10.1109/72.554196
- [28] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2015.7298682
- [29] Huang, G., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition, 2008*.
- [30] Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. Retrieved May 2, 2019, from <http://neuralnetworksanddeeplearning.com>
- [31] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved from <http://www.deeplearningbook.org>
- [32] Kingma, D. P., & Ba, J. L. (2014, December 22). Adam: A Method for Stochastic Optimization. Retrieved May 8, 2019, from <https://arxiv.org/abs/1412.6980>
- [33] VImage Programming Guide. (2016, September 13). Retrieved May 8, 2019, from <https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>
- [34] Hadsell, R., Chopra, S., & Lecun, Y. (n.d.). Dimensionality Reduction by Learning an Invariant Mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR06)*. doi:10.1109/cvpr.2006.100
- [35] Ma, W., & Manjunath, B. (1996). Texture features and learning similarity. *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.1996.517107
- [36] Lenail, A. (n.d.). NN-SVG. Retrieved May 14, 2019, from <http://alexlenail.me/NN-SVG/AlexNet.html>
- [37] TensorFlow. (n.d.). Retrieved May 15, 2019, from

<https://www.tensorflow.org>

- [38] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics*, 249-256.
- [39] Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica Et Biophysica Acta (BBA) - Protein Structure*, 405(2), 442-451. doi:10.1016/0005-2795(75)90109-9

TRITA-EECS-EX-2019:512