TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

XINGYANG NI

COMPETITIONS IN EDUCATION: CASE STUDY ON FACE VERIFICATION

Master of Science Thesis

Examiner: University Lecturer Heikki Huttunen
Examiner and Topic approved by the
Faculty Council of the Faculty of
Computing and Electrical Engineering
on Oct 7, 2015

ABSTRACT

XINGYANG NI: Competitions in Education: Case Study on Face Verification
Tampere University of Technology
Master of Science Thesis, 60 pages
April 2016
Master's Degree Programme in Information Technology
Major: Signal Processing
Examiner: University Lecturer Heikki Huttunen
Keywords: Face Verification, Competition, Deep Learning

All genuine knowledge originates in direct experience, especially for engineering courses. To help the students grasp hands-on experience of solving practical problems, a Machine Learning competition named *TUGraz-TUT Face Verification Challenge* was jointly organized by *Graz University of Technology* and *Tampere University of Technology*. The objective of the competition was to identify whether two facial images represent the same person. During the two-month period, the competition received 137 entries submitted by 28 players in 20 teams. This thesis summarizes the outcome of the competition.

To scrutinize the face verification system systematically, the processing workflow was divided into several parts. In the procedure of face alignment, *Unsupervised Joint Alignment* and *Ensemble of Regression Trees* were compared. Subsequently, the *OpenFace* and *VGG Face* features were retrieved from the aligned images. In the classification system, the performance of neural network and support vector classification were evaluated. Moreover, the influence of the ensemble strategies and the result of different error metrics were investigated. Based on the cutting-edge deep neural networks proposed by the research community, the winning solutions attained excellent results as the Weighted AUC scores exceeded 0.9990.

In addition to the preceding accomplishments, the findings suggested that there were still opportunities for further enhancements of the face verification systems. The limitations of current work and a handful of conceivable directions for future research had been deduced.

PREFACE

In late 2014, a online Machine Learning competition was organized as a compulsory assignment of one course offered by *Tampere University of Technology*. The winning solutions from that competition clearly surpassed the state of the art methods. Meanwhile, the feedback from the students was quite positive as they delighted in learning by doing. So as to solidify such good practice, we endeavor to integrate more competitions with the conventional classroom teaching.

This thesis subjects to another competition which was carried out in late 2015. To expand the scope of the participants, we cooperated with *Graz University of Technology*. It is plain fact that the Deep Learning techniques emerge as the optimum methods in many research areas of Computer Vision in recent years. We chose *Face Verification* as the content of the competition since the Deep Learning techniques has enabled the Machine Learning systems to approach human-level performance in recognizing faces.

I would like to express my sincere appreciation for my supervisor Heikki Huttunen who has always being willing to providing professional guidance whenever needed. I also wish to thank Gernot Riegler for collecting the valuable data set for the competition. Moreover, I am obliged to my colleague Andrei Cramariuc for the technical discussions about the possible solutions. In addition, I want to extend my sincere gratitude to the generous and munificent Finns for offering the opportunity to pursue my Master's degree in such an awesome country.

Last but not least, I am beholden to my family members for their tremendous support in every decision I have made.

Tampere, Finland

April 18, 2016

Xingyang Ni

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ACC	Accuracy
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Cross-Validation
DL	Deep Learning
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GD	Gradient Descent
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
LMNN	Large Margin Nearest Neighbor
MCC	Matthews Correlation Coefficient
ML	Machine Learning
MLP	Multilayer Perceptron
MOOC	Massive Open Online Course
NN	Neural Network
NPV	Negative Predictive Value
PCA	Principal Component Analysis
PPV	Positive Predictive Value
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SVC	Support Vector Classification
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate

1. INTRODUCTION

In 1959, Arthur Samuel developed a program which plays the game of checkers and it is regarded as the first-ever self-learning program. The Samuel Checkers-playing Program defeated the champion in the State of Connecticut two years later. [57, 58] In 2016, a program named *AlphaGo* sealed 4-1 victory over the Go Master Lee Sedol. Go is a board game which depends heavily on intuition and the number of possible legal positions far exceeds the total amount of atoms in the observable universe. [31, 60, 64] Machine Learning (ML) makes these two programs possible. It is also the enabling technique of speech recognition, off-road autonomous driving and targeted display advertising [21, 52, 67]. As a result of the exponential growth of computing power and the flood of enormous data, Machine Learning is booming throughout the years. However, imparting the learning aptitude to computers remains to be the most challenging objective in Artificial Intelligence (AI) [44].

Many Massive Open Online Course (MOOC) platforms emerge as a popular alternative to the traditional education since 2012, such as *edX*, *Udacity* and *Coursera*. [49] These service providers often collaborate with the prestigious universities and leading companies all over the world. They offers a precious opportunity for prospective students to obtain new skills and knowledge which were unreachable to them in the past. As the importance of web-based learning could not be overemphasized, a Machine Learning competition was jointly organized by *Graz University of Technology* and *Tampere University of Technology*. The students from both universities were induced to take part actively.

Faces have irrefutably huge influence on people's opinions about the first impressions of a stranger. [74] Since faces are important in social interactions, it has been a popular topic in the Machine Learning community. Typical research areas includes face detection, face alignment, face verification and recognizing facial expressions. [4, 9, 15, 75] Among these research areas, face verification is one of the most challenging problems and the objective is to verify whether two images represent the

same person. The humans have astonishing abilities to identify familiar faces instantly even after years of separation. However, the same task introduces a bunch of tricky problems to the computers since there could be different viewing conditions, expressions and distractions. The machine learning algorithm need to be capable of ignoring nuisance variables and concentrating on pertinent features. [35, 72] In the competition, approximately 2700 human facial images were retrieved from the Internet. Each image belongs to a specific person and each person has roughly 5 images on average. The image data set was evenly divided to two groups, namely, the training data set and the testing data set. While the matching relation between the image and the person in the training data set was revealed, the contestants were required to predict whether each possible pair of the images in the testing data set were collected from the same person.

The remainder of the thesis is structured as follows. Section 2 presents the theoretical background. Section 3 formalizes the content of the competition and introduces the solutions from two top performing teams along with other available approaches. The performance of the learning system is analyzed comprehensively. Finally, the conclusions and possible future work are discussed in Section 4.

2. THEORETICAL BACKGROUND

2.1 Machine learning

Various definitions could be found for the discipline of machine learning. In 1959, Arthur Samuel originated the notion: "field of study that gives computers the ability to learn without being explicitly programmed". [65] In 1997, Tom Mitchell described it as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E". [45] The subject of machine learning seeks out the answer for how to construct an intelligent computer system which could improve with experience spontaneously and what are the ground rules that pertain to all learning processes. [46] Depending on the intrinsic property of the provided data set, the learning systems could be distinguished into the following three board categories.

Firstly, supervised learning refers to those scenarios where the desired output is available. The training data set contains a certain amount of examples and each example consists of the feature vector along with the corresponding desired output value. A supervised learning algorithm explores the causality between the feature vector and the desired output value. Supervised learning problems could be further divided into two types. On the one hand, the output might represents different classes, and therefore the value should be discrete. Such problems are regarded as classification problems. For example, forecasting whether it will rain falls under this category. On the other hand, the output value could be continuous rather than discrete. These cases are considered to be regression problems. For instance, predicting the outdoor temperature belongs to this group. [56]

Secondly, unsupervised learning refers to those scenarios where the desired output is absent. As a consequence, the conventional evaluation methods are not applicable and an unsupervised learning algorithm is expected to find out the hidden structures

inside the unlabelled data set. [56] An e-commerce service provider could collect the data set which characterize the customers and apply unsupervised learning techniques to map each customer to several clusters. An interest-based recommendation system could be implemented based on the outcome of the unsupervised learning algorithm.

Thirdly, reinforcement learning focuses on maximizing the cumulative reward by learning what actions to take. Instead of explicitly feeding the correct input and output pairs, the learning system need to figure out which actions could bring in the most reward by trying them. Reinforcement learning differs from supervised learning as a reinforcement learning system learns from interaction while a supervised learning system learns from examples supplied by external knowledgeable supervisors. Consider a robot vacuum cleaner, it needs to make a choice between cleaning another room or returning to the charging station. The optimal decision could be made by reviewing how readily it has been able to find the charging station in the past. Such interaction with the surroundings is indispensable for adapting behaviours. [68]

2.2 Face verification

Most conventional face verification systems only work when the input image is frontal and those systems fail catastrophically if the facial image is captured from non-frontal viewpoints. [7] Although manipulating the non-frontal facial images remains difficult, this thesis focuses on the state of the art solutions which utilized Deep Learning techniques to perform face verification.

Figure 2.1 demonstrates the general workflow of a modern face verification system named *OpenFace*. Firstly, the system detects the faces in the given input image. Secondly, the system transforms the original facial image and generates an aligned facial image. This face alignment process is considered to be necessary as it helps to improve the overall performance [66]. Thirdly, the aligned facial image is passed to a deep neural network which is designed to extract an informative feature vector of the face. In the ideal case, a smaller distance between two feature vectors signifies that the faces in these two images are more likely to be the same person. Last but not least, one may exploit clustering, similarity detection and classification methods to accomplish a fully functional face verification system. [2]

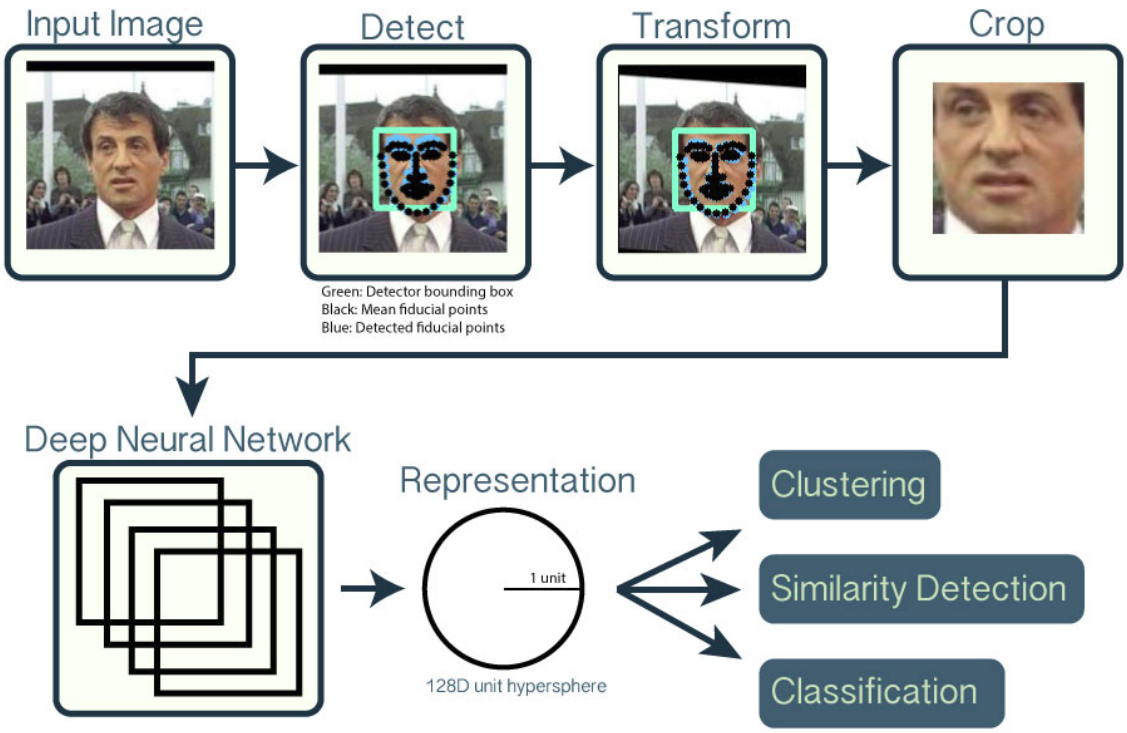


Figure 2.1 *General workflow of a modern face verification system named OpenFace [2].*

2.3 Face alignment

Performing face alignment on the images in the unrestrained conditions is still a problematic task. As a human face is a 3D object, it naturally has different poses. In addition, there are various kinds of facial expressions, such as happy, surprised, angry and so on. Many complicated methods have been purposed to resolve these limitations. Typical face alignment algorithms fall into the following three broad categories. Some schemes fit a 3D model based on the facial image. Some schemes find similar fiducial-points configurations from an external dataset. Some schemes apply unsupervised methods which search for a similarity transformation based on the pixel values. [70] In this section, two face alignment algorithms which have been applied in the competition are explained.

2.3.1 Unsupervised joint alignment

Gary Huang *et al* proposed an alignment mechanism which employs poorly aligned examples of faces with no additional labels. The congealing process has been applied

to the images in the training date set. Initially, the empirical distribution field of these images is calculated. After that, the algorithm finds a suitable transformation which maximizes the likelihood of each image according to the distribution field. Subsequently, the distribution field is updated according to the transformed images. The aforesaid steps are repeated until convergence. In order to align additional images, the distribution fields from each iteration step are recorded. With the intention of adapting the basic congealing algorithm to work on complex images, the SIFT descriptors [41] each pixel are processed by using k-means clustering algorithm. Therefore, the distribution fields consist of distributions over the possible clusters of each pixel. Moreover each pixel is treated as a mixture of the underlying clusters and it helps to suppress the local minima problems. This novel method could increase the performance of a face recognizer as the original faces are transformed into a canonical pose. [25]

2.3.2 Ensemble of regression trees

Vahid Kazemi *et al* devised an algorithm which is capable of detecting the facial landmarks from a single image in one millisecond. In order to estimate the coordinates of the facial landmarks, an ensemble of thousands of shallow regression trees was trained with 2000 facial images. Each image was warped with 20 different initial guesses for the face's shape, thus the training data set was extended by a factor of 20. The squared error loss function was minimized by using the gradient boosting algorithm in each regressor. As the gradient boosting algorithm might suffer from the overfitting problem, different regularization strategies had been analysed. The result showed that the averaging regularization and the shrinkage method could effectively reduce the variance by learning multiple overlapping models. In term of accuracy, this algorithm reaches or exceeds the state of the art methods on the standard data sets. [29] Based on the detected landmarks, one could apply an affine transformation to make the eyes and bottom lip appear in the same location on each image [2].

2.4 Neural networks

Motivated by the mechanism of the human brain, Artificial Neural Network (ANN) could perform useful computations through a learning process [22]. In 1943, Warren

McCulloch *et al* presented a logical calculus which involves in nervous activities [42]. In 1958, Oliver Selfridge proposed a process named *Pandemonium* which could adaptively improve itself to solve certain tasks [62]. In the same year, Frank Rosenblatt demonstrated the capabilities of the perceptron [43]. In 1980, Kunihiko Fukushima created a multilayered structure which is capable of unsupervised learning [19]. In 1986, David Rumelhart *et al* revealed the significance of the backpropagation algorithm as it worked much faster than earlier methods in neural networks [78]. In 1998, Yann LeCun *et al* designed a Convolutional Neural Network (CNN) which could recognize handwritten and machine-printed characters [37]. In 2012, Alex Krizhevsky *et al* applied a CNN to solve the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and the result was significantly better than the previous approaches [16, 34].

At the early days, neural network techniques were constrained by low computing power and tiny data sets. The size of the neural network was relatively small because the computers were unable to perform a lot of computations. As a consequence, other algorithms such as Support Vector Machine (SVM) outperformed neural networks in many Machine Learning applications. [59, 76] In the last few years, neural networks regained popularity in the research community since it made great breakthrough in speech recognition, computer vision and machine translation [13, 21, 34]. The potential of neural networks was unleashed by large amount of data and ultrafast Graphics Processing Units (GPUs) [54]. To differentiate the neural networks, Deep Learning (DL) is a buzz word which refers to neural networks with lots of nonlinear layers. Nowadays, extremely deep neural networks are reachable. The researchers from Microsoft implemented neural networks which consisted of 152 layers and they secured first place in ILSVRC 2015 [23]. Among the deep neural networks, the convolutional neural network is the dominant approach to many computer vision problems [34]. In the following subsections, the essential theorems of convolutional neural networks and the face descriptors in *OpenFace* and *VGG Face* are explained.

2.4.1 Neurons in regular neural networks

A neural network is a graph of connected neurons. The mathematical model of a neuron is shown in Figure 2.2. The input signals $x = x_0, x_1, ..., x_n$ propagate along the axons and interact multiplicatively with the dendrites $w = w_0, w_1, ..., w_n$ of another neuron in an element-wise manner. The values of w refer to the strengths of the connection between two neurons. In the cell body, the intermediate output

is denoted by $v = \sum_i w_i x_i + b$ and it is further passed to an activation function f which generates the output signal. The activation function f is applied to model the firing rate of the neuron. Several activation functions are analyzed in the following paragraphs. [1, 73]

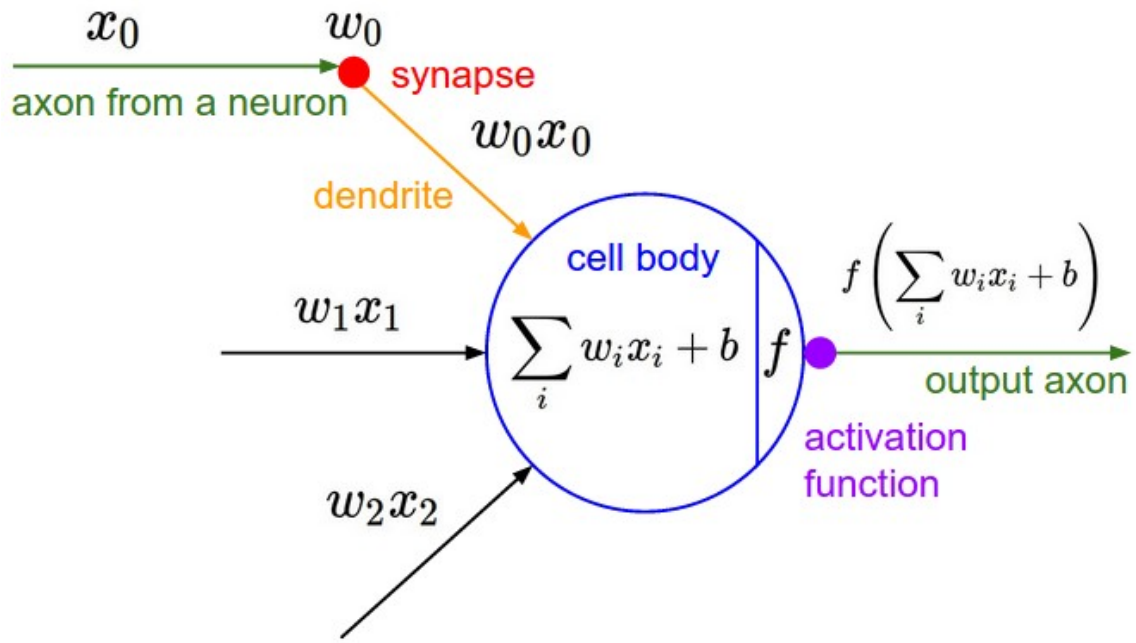


***Figure** 2.2 Mathematical model of a neuron [73].*

Each activation function performs a mathematical operation on the input value. The definitions and visualizations of four activation functions are shown in Table 2.1 and Figure 2.3. Firstly, the sigmoid function maps the input value to the interval between 0 and 1. To be more specific, large negative numbers get 0 while Large positive numbers get 1. For the reason that sigmoid saturates gradients and the average of the outputs is not zero, the sigmoid function is rarely used in practice. Secondly, the tanh function is a scaled variant of the sigmoid function. Because the output of tanh is zero-centered, the tanh function is always more desirable than the sigmoid function. Thirdly, Rectified Linear Unit (ReLU) differentiates the inputs at 0. On the one hand, the negative inputs get 0. On the other hand, the outputs are the same as the inputs if the inputs are positive. The ReLU function could not only boost the convergence speed but also decrease the computational burden. However, some neurons might be never activated for the entire training data set on the condition that the learning rate is too high. This defect could be suppressed by selecting a proper learning rate. Finally, Parametric Rectified

Linear Unit (PReLU) is introduced to improve the learning capability of the neural networks. An additional parameter is added to the regular ReLU activation function and the parameter a controls the slope of the negative inputs. One study showed that PReLU could achieve better performance than ReLU. [24, 73]

Table 2.1 *Definitions of activation functions [24, 73].*

Activation function	Definition
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$ (2.1)
Tanh	$f(x) = \frac{2}{1+e^{-2x}} - 1$ (2.2)
ReLU	$f(x) = max(0, x)$ (2.3)
PReLU	$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x \leq 0 \end{cases}$ (2.4)

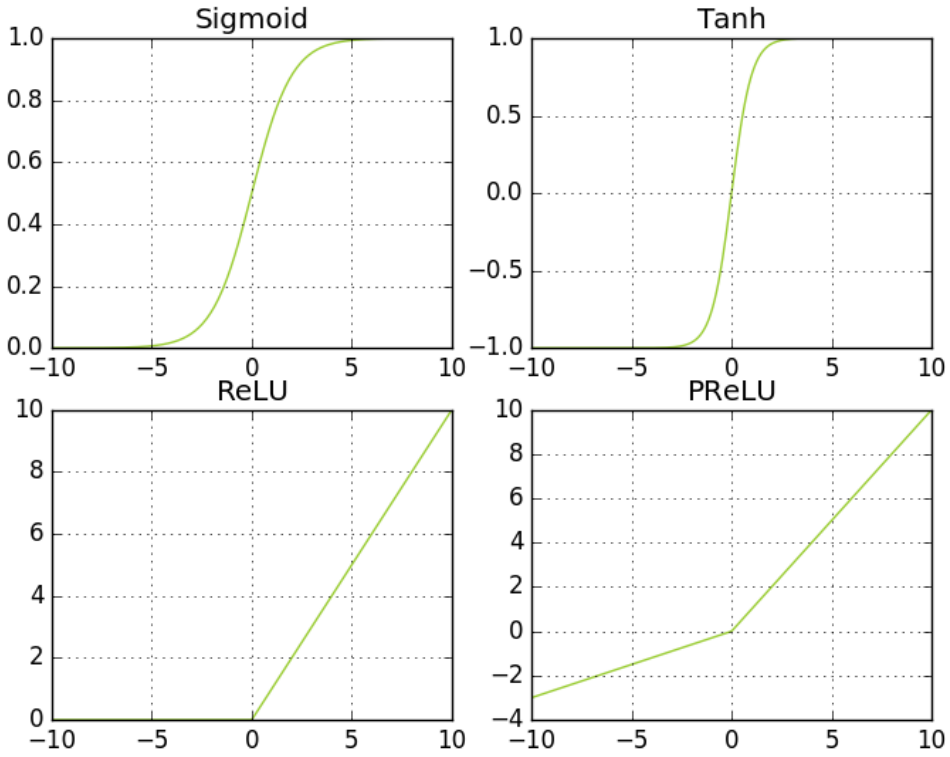


Figure 2.3 *Visualizations of activation functions [24, 73].*

2.4.2 Neurons in convolutional neural networks

A regular neural network takes a feature vector as the input and manipulates it through a sequence of layers. Figure 2.4 illustrates the structure of a regular 3-layer neural network. The neurons in the hidden layers are fully connected to all neurons in the previous layer. Meanwhile, the neurons in the same layer are independent from each other. However, this fully-connected configuration is inappropriate for images. Suppose the input color image has 200 rows and 200 columns, the size of input is $200 \times 200 \times 3$. A single neuron in the first hidden layer would have $200 \times 200 \times 3 = 120000$ weights. Consequently, the regular neural network is wasteful of the limited computing power. [73]

Convolutional neural networks make use of the fact that the inputs are images. The neurons are organized in 3 dimensions: width, height and depth (see Figure 2.5). The neurons in one layer will only be connected to the neurons in a small region of the previous layer. The hyperparameter *receptive field* constrains the spatial extent of connectivity. The hyperparameters *depth*, *stride* and *zero-padding* control the size of the output volume. Moreover, the parameter sharing scheme is utilized to sharply reduce the total amount of parameters in the convolutional neural networks. The neurons in each depth is processed by using the same weights and bias. [73]

2.4.3 Layers in convolutional neural networks

Convolutional layers, pooling layers and fully-connected layers are common and popular in convolutional neural networks. Figure 2.6 presents an example where these layers are stacked on top of each other. The convolutional layers calculate the output of a neuron in one layer by using the output of the neurons in a small region of the previous layer. Afterwards, the pooling layers downsample the width and height dimensions while the depth dimension remains unchanged. The convolutional and pooling layers might be repeated for several times. Finally, the fully-connected layers are appended at the end. The neurons in the fully-connected layers take the output of all the neurons in the previous layer as the input signals. In general, the convolutional neural networks provide an end-to-end solution which maps the input images to corresponding labels. Instead of feeding the low-level handcrafted features, the convolutional neural networks could learn the high-level features autonomously. [63, 73]

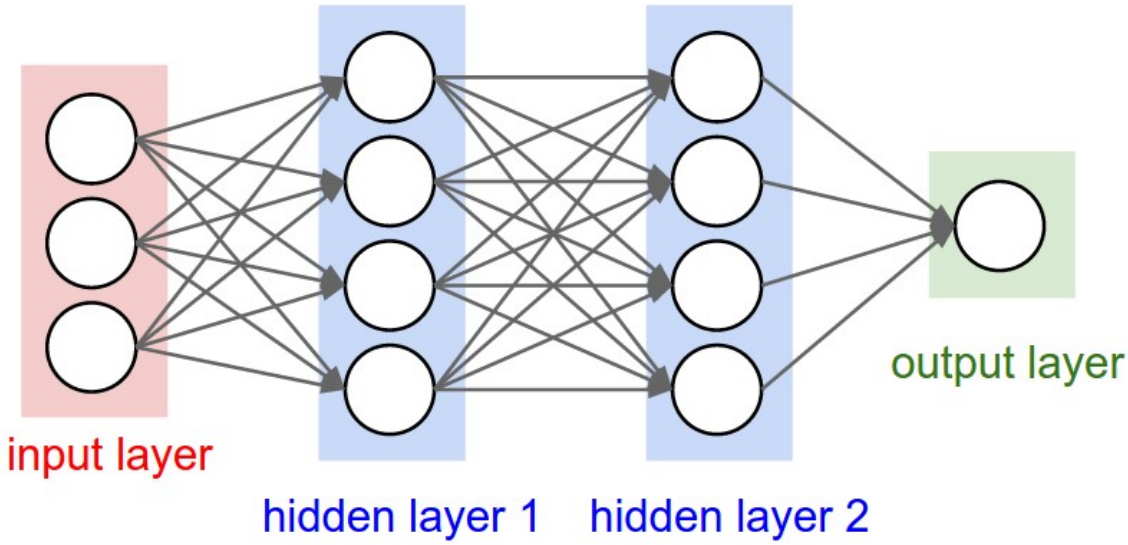


Figure 2.4 *The structure of a regular neural network [73].*

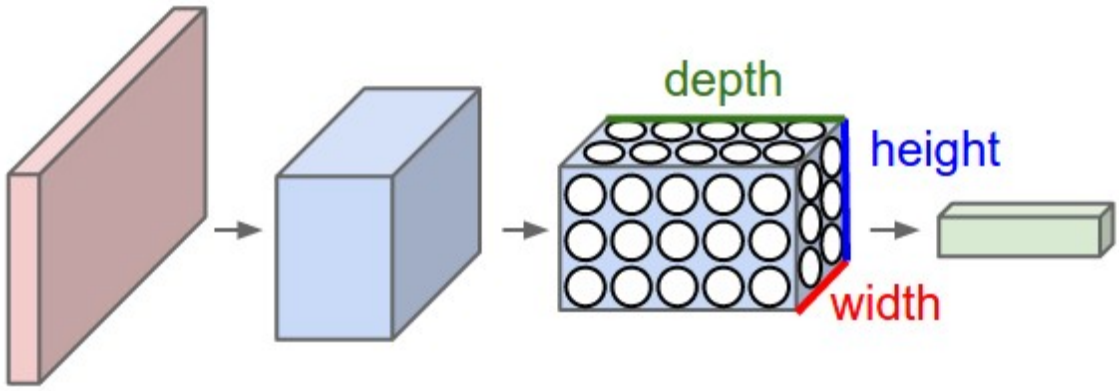


Figure 2.5 *Neurons in convolutional neural networks [73].*

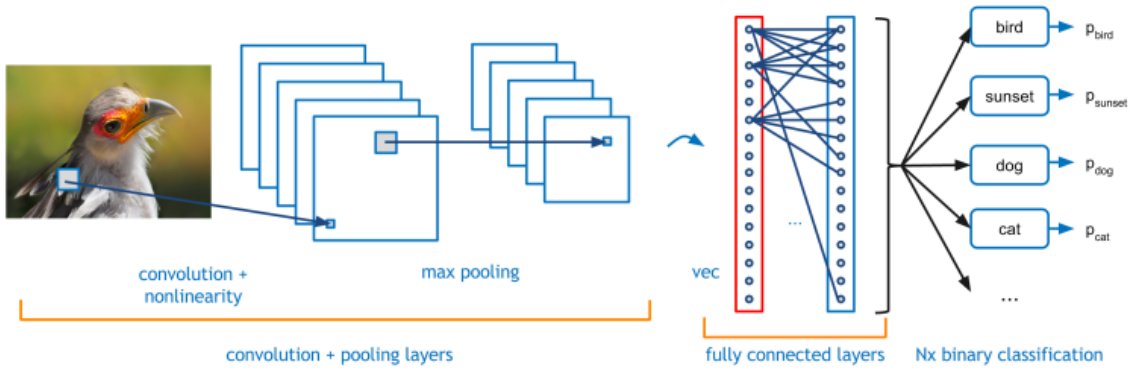


Figure 2.6 *The structure of a convolutional neural network [63].*

2.4.4 Optimization of neural networks

In the training phase of a neural network, the parameters inside the network need to be optimized interactively. Based on the differences between the desired and actual outputs, backpropagation combined with suitable optimization methods could be applied to update the coefficients. Many optimization algorithms have been devised, such as AdaGrad, Adadelta, Adam and RMSprop [17, 30, 71, 83]. In this subsection, the basic ideas behind gradient-based methods are explained.

In a supervised learning problem, each observation z could be denoted by a pair (x, y) where x is an arbitrary input and y is a scalar output. A loss function $\mathscr{L}(y, \hat{y})$ evaluates the discrepancy between the ground truth value y and the predicted value $\hat{y}$. Given a set of parameters w, the predicted value is determined by $\hat{y} = f_w(x)$. Thus, one need to minimize the loss $Q(z, w) = \mathscr{L}(y, f_w(x))$. For n observations indicated by $z_1, z_2, ..., z_n$, the empirical risk is: [8]

$$E_n(f_w) = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(y_i, f_w(x_i)) \tag{2.5}$$

In the method of Gradient Descent (GD), the parameters w is updated based on the gradient of $E_n(f_w)$ as shown below: [8]

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \bigtriangledown_w Q(z_i, w_t) \tag{2.6}$$

where γ is a suitable gain.

By contrast, the Stochastic Gradient Descent (SGD) is a simplified variant of Gradient Descent. Instead of getting an accurate $E_n(f_w)$, the gradient is estimated based on single randomly chosen observation z_t: [8]

$$w_{t+1} = w_t - \gamma_t \bigtriangledown_w Q(z_t, w_t) \tag{2.7}$$

SGD is more appropriate than GD for several reasons. Firstly, SGD is usually much faster than GD particularly on the data sets which contain large amount of redundant records. Secondly, SGD could achieve better solutions in most cases. Since

the estimated gradient is computed by using only one observation, the parameters w might not be updated accurately along the actual gradient. The noisy estimated gradient contributes to avoiding local minimums. Thirdly, SGD works better in an online learning problem. As the new data set comes in, the changes in the data stays unnoticed with the GD algorithm. This would lead to large residuals in the predictions. [38]

2.4.5 Face descriptor in *OpenFace*

OpenFace is an open-source replication of *FaceNet* which was originated by the researchers from Google. While *FaceNet* was trained with proprietary data set which had 200 million facial images from 8 million different identities, *OpenFace* was trained with 0.5 million facial images from 10 thousands different identities by coalescing the CASIA-WebFace [82] and FaceScrub [47] data sets. [2, 61]

The customary face verification systems applied a classification layer to identify the each individual in the data set and the feature vectors were retrieved from an intermediate layer. However, this intermediate layer could be a bottleneck as the representation might not generalize well to the unseen faces and the dimensionality of the feature vectors was typically quite high. To address this challenge, the convolutional network was directly trained to optimize the triplet loss. Figure 2.7 visualizes the learning procedure of triplet loss. A triplet consists of three elements. The *Positive* and *Anchor* elements belong to the same person while the *Negative* element represents another person. The learning procedure strives to make the *Positive* element closer to the *Anchor* element than the *Negative* element. [61]

Figure 2.7 *Learning procedure of triplet loss [61].*

OpenFace borrowed the network configuration defined in *FaceNet* (see Table 2.2). Since the available data set was much smaller, *OpenFace* reduced the size of the

neural network in order to improve the computational efficiency. The inception layers increased the depth and width of the neural network without requiring additional computing resources [69]. The CNN was trained by using SGD with backpropagation [36, 55] and Adagrad [17]. The learning rate was decreased at the final stage of the training phase. [2, 61]

Table 2.2 *Network configuration in FaceNet [61].*

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

2.4.6 Face descriptor in *VGG Face*

In the work of *VGG Face*, Omkar Parkhi *et al* presented a convolutional neural network which achieved analogous state of the art results on the *Labeled Faces in the Wild (LFW)* [26] and *YouTube Faces (YTF)* [79] benchmark data sets. A multi-stage strategy was applied in the data collection process and over 2.6 million images were gathered from 2622 identities with limited amount of manual annotation. This data set contained the largest amount of facial images among the publicly available data sets. [50]

To begin with, the problem was regarded as a classification task whose objective was to discriminate $N = 2622$ unique individuals. The final fully-connected layer in the CNN contained N linear predictors. Each linear predictor corresponds to an identity in the training data set. Given an input image, the output scores of these linear predictors were compared to the ground truth identity by using the empirical softmax log-loss. After minimizing this loss function during the training phase, the

input of the final fully-connected layer was a concise representation of the facial image. The retrieved feature vectors and the triplet loss training scheme could be exploited in the face verification systems. This bootstrapping scheme relieved the complexity of the training phase considerably. [50]

Table 2.3 shows a network configuration in *VGG Face*. The neural network was considered to be very deep since it contained a long sequence of convolutional layers. The weights of the filters in the CNN were initialized by random sampling from a Gaussian distribution while the Biases were initialized to zero. The optimization was achieved by using SGD with momentum. The overfitting problem was suppressed by applying dropout and weight decay. After the accuracy of the model on the validation data set stopped increasing, the learning rate was decreased by factor of 10. [50]

Table 2.3 *Network configuration in VGG Face [50].*

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	softmx
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	relu7	fc8	prob
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

2.5 Error metrics

In a machine learning task, one needs to evaluate the performance of a specific solution. An error metric is applied to distinguish a good solution from the bad solutions. Different error metrics may lead to different ranks of several possible methods. Unfortunately, there is no single error metric which is suitable to any machine learning task. As a consequence, one need to select an error metric that is appropriate for current task. Ideally, the error metric should be reliable so that a method which achieves a better score should be a better solution in practical sense.

2.5.1 Confusion matrix

In a binary classification problem, the algorithm detects whether an event takes place or not. The prediction for a sample is either true or false. A confusion matrix visualizes the performance of a binary classifier. Each row refers to the samples in an actual class, while each column refers to the samples in a predicted class (or vice-versa) [53]. As shown in Table 2.4, the predictions are divided into four categories, namely, True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN). The term positive/negative refers to the predicted class while the term true/false refers to whether the predicted class is the same as the actual class. It is apparent that one should try to increase the occurrences of true positive/negative and decrease the occurrences of false positive/negative. In the ideal case, the number of occurrences of false positive/negative could be 0 which means that no sample is misclassified. Suppose one need to discern the category of animals and the testing data set contains images from dogs and cats, a classifier detects 50 dogs among 60 dogs and 20 cats among 40 cats. Assume that the prediction value true denotes a dog while the prediction value false denotes a cat, the number of occurrences of TP, FN, FP and TN are 50, 10, 20 and 20, respectively.

Table 2.4 *Confusion matrix for a binary classification problem [40, 77].*

		Predicted Class	
		+	-
Actual Class	+	True Positive (TP), Hit	False Negative (FN), Miss
	-	False Positive (FP), False Alarm	True Negative (TN), Correct Rejection

2.5.2 Common error metrics derived from confusion matrix

Based on the confusion matrix, one can retrieve four numbers which correspond to the occurrences of four categories. However, it is unintuitive to compare the performance of two solutions by using these four numbers directly. With the intention of describing the performance of a solution with only one number, several error metrics have been derived from confusion matrix (see Table 2.5). Each error metric focuses on an aspect of model assessment. True Positive Rate (TPR) refers to the proportion of predicted positive items among actual positive items. It measures how good the method is in detecting positive items. Conversely, False Positive Rate (FPR) is defined as the ratio of predicted positive items among actual negative

items. It gauges how bad the method is in detecting negative items. Furthermore, Positive Predictive Value (PPV) denotes the percentage of actual positive items among predicted positive items. It appraises the reliability of the positive predictions. Contrariwise, Negative Predictive Value (NPV) corresponds to the fraction of actual negative items among predicted negative items. It evaluates the trustworthiness of the negative predictions. Last but not least, F Measure is a combination of precision and recall while Accuracy (ACC) calculates the percentage of correct predictions. [53]

Table 2.5 *Common error metrics derived from confusion matrix [18, 53].*

Error Metric	Definition
True Positive Rate (TPR), Sensitivity, Recall	$TPR = \frac{TP}{TP+FN}$ (2.8)
False Positive Rate (FPR), Fall-out	$FPR = \frac{FP}{FP+TN}$ (2.9)
Positive Predictive Value (PPV), Precision	$PPV = \frac{TP}{TP+FP}$ (2.10)
Negative Predictive Value (NPV)	$NPV = \frac{TN}{TN+FN}$ (2.11)
F Measure	$F_\beta = \frac{(1+\beta^2)TP}{(1+\beta^2)TP+\beta^2FN+FP}$ (2.12)
Accuracy (ACC)	$ACC = \frac{TP+TN}{TP+FP+TN+FN}$ (2.13)

TPR, PPV and F Measure put emphasis only on the positive cases. None of them captures any information about how well the solution tackles the negative cases. [53] In an unbalanced data set, the vast majority of the samples are labelled as one class, while far less samples are labelled as another class. Such unbalanced data sets could be found in many real-world domains, such as detection of fraudulent telephone calls and learning word pronunciations. [33] Due to the dominating effect of the majority class, the conventional way of maximizing overall accuracy often fail to learn anything meaningful about the minority class [77].

2.5.3 Matthews correlation coefficient

Matthews correlation coefficient (MCC) is another error metric derived from confusion matrix and it is generally considered to be an appropriate evaluation method

even if the data set is heavily unbalanced. MCC takes all four categories (TP, TN, FP, FN) into account and it can be written as: [53]

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.14}$$

Essentially, the MCC is a correlation coefficient between the ground truth and prediction. The value of MCC varies from -1 to $+1$. -1 indicates total disagreement and $+1$ represents total agreement. A completely random prediction will yield a coefficient of 0. [3]

2.5.4 Receiver operating characteristic

Receiver operating characteristic (ROC) curve is a popular evaluation method in the research area of Data Science. As the discrimination threshold changes, the curve is obtained by plotting the true positive rate against the false positive rate. With a lower discrimination threshold, a machine learning model tends to retrieve more positive samples, but also misclassify more negative samples as positive samples. [77] Both TPR and FPR increase in this case. Correspondingly, the ROC curve is a monotonically increasing function. Suppose the discrimination threshold is positive infinity, all samples are classified as false. TP and FP are 0. Hence TPR and FPR are 0. On the contrary, TPR and FPR are 1 if the discrimination threshold is negative infinity.

In general, a ROC curve is considered to be superior if it is closer to the upper-left corner. Area under the curve (AUC) is a common statistic could be applied to recapitulate a model's performance into a single scalar value. [77] Since the ROC curve of a random prediction is a diagonal line connecting $(0, 0)$ and $(1, 1)$, the AUC score is 0.5. Any reasonable solution should achieve an AUC score higher than 0.5. [18]

Figure 2.8 consists of two ROC curves. Suppose each point in the Curve A is denoted by (x, y), then the Curve B is generated by plotting $(1 - y, 1 - x)$. One could plainly see that the conventional AUC of Curve A and Curve B is exactly the same. In applications such as blocking scam websites, the search engines should only blacklist the websites under the condition that those websites are highly likely to be dangerous. Blocking a normal website by mistake may even lead to serious

lawsuits. In other words, a false alarm is much more costly than a miss. One should strive to maximize the TPR when the FPR is still low. Curve B is more suitable than Curve A. In other applications such as detecting cancer, the patients do not want to loss the opportunity to treat the disease in advance. A miss is much more detrimental than a false alarm. One should endeavour to minimize the FPR when the TPR is high enough. Curve A is more appropriate than Curve B.

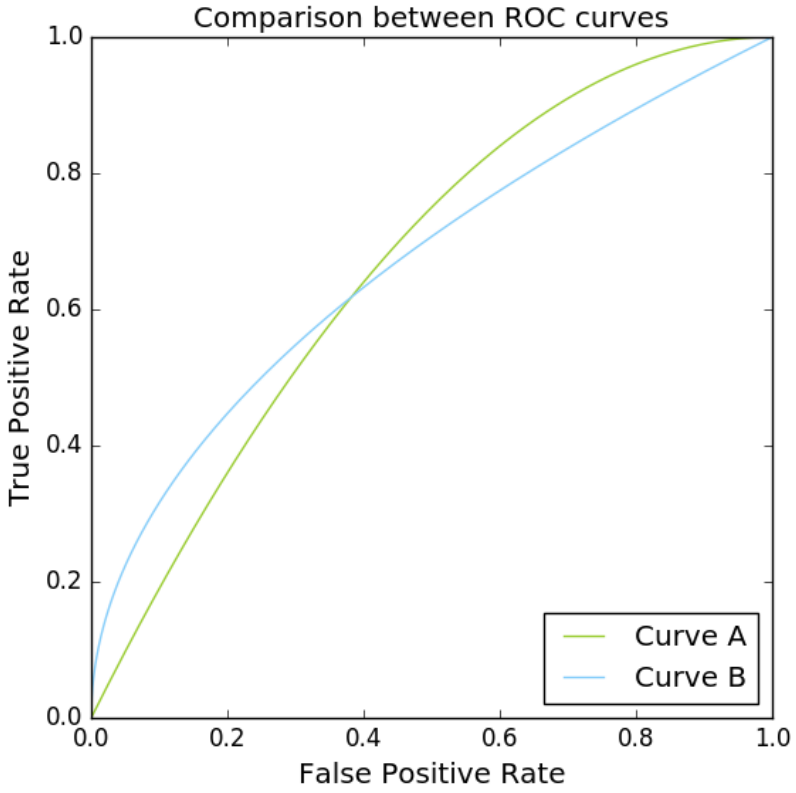
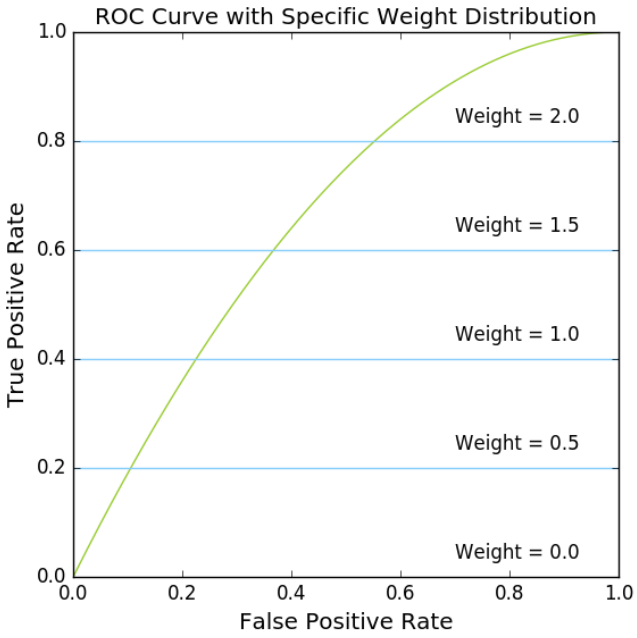
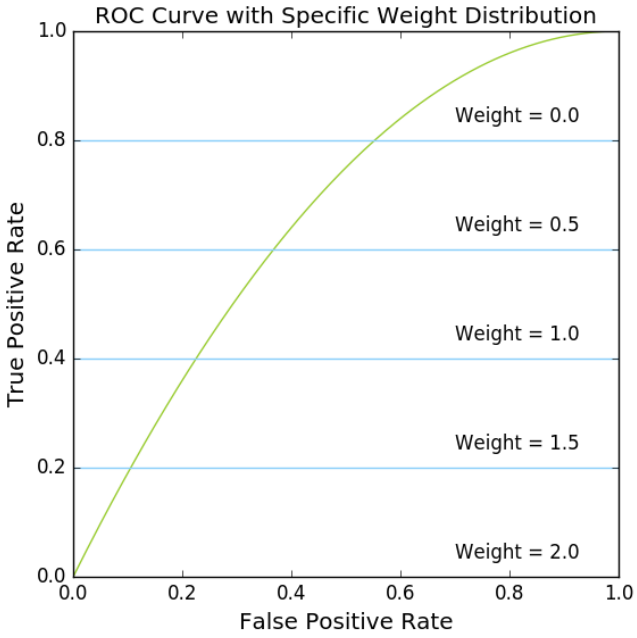


Figure 2.8 *Comparison between ROC curves when conventional AUC is the same [77].*

Due to the nature of the unbalanced data set, the misclassification cost of false alarm and miss could be quite different from each other. However, the conventional AUC does not take the uneven misclassification cost into account. The method of Weighted AUC is proposed to overcome such limitation. The square is divided into several rectangles along the TPR axis. The weight distribution of each rectangle is skewed based on the application. Figure 2.9 illustrates two possible weight distributions. If a miss is more damaging than a false alarm, more weight should be allocated to the rectangles with higher TPR. The weight distribution shown in Figure 2.9(a) is more appropriate. [77] If a false alarm is more damaging than a



(a) ROC curve with specific weight distribution which emphasizes the role of high TPR.



(b) ROC curve with specific weight distribution which emphasizes the role of low TPR.

Figure 2.9 *Comparison between weight distributions.*

miss, more weight should be allocated to the rectangles with lower TPR. The weight distribution shown in Figure 2.9(b) is more suitable. In addition to Weighted AUC, one may compare the value of TPR at a low FPR, such as 0.01. This criterion only involves one specific point in the ROC curve. If a learning system obtains higher TPR at the same FPR, this learning system could detect more True Positive examples and it is considered to be superior than other learning systems.

2.6 Cross-validation

A learning algorithm usually contains some hyperparameters which could not be automatically optimized and one need to define them before the actual training phase. The procedure of finding the ideal hyperparameters which achieve the best result for a specific problem is called hyperparameter optimization. In practice, one could define the space of hyperparameters and generate different combinations of those hyperparameters by using randomized search or grid search. Cross-validation (CV) plays an essential role in hyperparameter optimization and it could be employed to assess the performance of each hyperparameter set solely with the original training data set. [5, 6, 51] With appropriate cross-validation strategy, one should be able to recognize the most suitable model for the current task. A model which achieves satisfactory results on the validation data set is deemed to obtain comparable outcome on the testing data set.

2.6.1 *k*-fold cross-validation

k-fold cross-validation is a conventional strategy among the cross-validation methods. While k refers to the number of folds, Figure 2.10 demonstrates one iteration of a 5-fold cross-validation. The original training data set consists of 30 observations which are symbolized by colour balls. The green and red balls denote True and False cases respectively. The original training data set is randomly partitioned to 5 mutually exclusive folds with approximately equal number of observations. Each fold could be retained as the validation data set while the remaining 4 folds works as the training data set. The learning algorithm is fed by the training data set and evaluated by the validation data set. The aforementioned training and evaluation phases are repeated 5 times and all the folders have been used as the validation data set exactly once. [32]

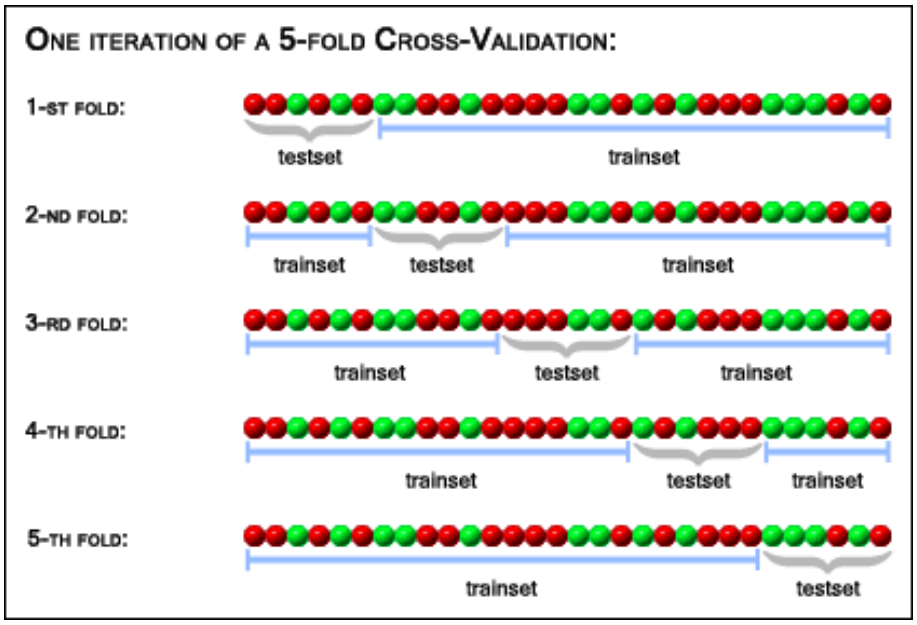


Figure 2.10 *One iteration of a 5-fold cross-validation [48].*

2.6.2 Variations of *k*-fold cross-validation

In addition to k-fold cross-validation, several variations have been devised. On the one hand, it is possible that the minority classes only exists in the validation data set in a heavily unbalanced data set. In other words, the learning algorithm might has never seen these minority classes at all in the training phase. Stratified k-fold cross-validation is an effective approach to address this limitation. The difference is that the original training data set is split in a stratified fashion. Accordingly, the resulting training and validation data set contain roughly the same proportions of labels as the original training data set. On the other hand, the data set may contain observations with respect to the year of collection and one might be interested at splitting the data set against the time information. In Leave-p-Label-Out cross-validation, the validation data set is built with all the observations which have one of the selected p different labels. [51]

3. RESULTS AND DISCUSSION

3.1 Overview of *TUGraz-TUT Face Verification Challenge*

Many organizations do not possess the state of the art Machine Learning techniques to solve their problems. In the meantime, the data scientists are in great need of the real-world data set to deepen their skills. *Kaggle* provides a platform to link the organizations which have both the data set and the problems to the data scientists who know about the possible solutions. To make the competitions more appealing and interesting, the organizations usually provide a certain amount of cash or job positions as reward. The participants with different backgrounds and specializations apply various kinds of methods to address the same challenge. Such competitive structure heartens the participants to triumph over each other and the winning teams are likely to outperform the existing best approach. Additionally, *Kaggle* offers self-service competitions for the academic institutions without charge. The university students have been granted a precious opportunity to grasp hands-on experience as part of their studies. [28]

The *TUGraz-TUT Face Verification Challenge* was organized from 14 October 2015 to 11 December 2015 on *Kaggle*. A group of students from *Graz University of Technology* and another group of students from *Tampere University of Technology* were encouraged to participate in this competition. Unlike many other competitions, those *Kaggle* users who were not students from the organizing universities were also allowed to participant. The top performing teams were awarded T-shirts as prize. [27]

In this competition, the training data set contained 1393 images while the testing data set contained 1343 images. Each image consisted of the face of a person. In the training data set, the images from the same person were placed in the same sub-folder. On the contrary, all the images in the testing data set were saved in single directory. Those images were originally collected by the colleagues from *Graz*

University of Technology. A Python script was implemented to crawl images of the celebrities from the Internet automatically. A manual sanity test was appended in order to avoid preposterous mistakes. Figure 3.1 illustrates an image pair selected from the training data set. It is apparent that these two images were taken from real life scenarios. There are huge differences in term of background, lighting condition and facial expression.

(a) Original image (*train_ 00000180.jpg*).

(b) Original image (*train_ 00000183.jpg*).

Figure 3.1 *An image pair selected from the training data set.*

Several additional properties of each image had been calculated beforehand and provided to the participants [27]. Firstly, The coordinates which define the bounding box around the face were computed by using a multi-view detection approach [81]. Secondly, the landmark positions of the face were estimated directly from a sparse subset of pixel intensities by using an ensemble of regression trees [29]. Thirdly, the deep features were extracted from the last hidden layer of *AlexNet* [34]. With the purpose of helping the participants to get started, a sample solution was given both in Matlab and Python. The difference between two images were described by using the Euclidean distance between two *AlexNet* feature vectors. Under the assumption that the smaller the distance the more likely the two pictures contain the same person, the Euclidean distance is converted to the final prediction value by inverting and normalizing. [27]

The participants were excepted to predict whether each possible image pairs in the testing data set contain the same person or not. As a result, more than 99.7% pairs do not contain the same person. The categorization accuracy measures the percentage of correct predictions. If one naively predict that all the pairs do not represent the same person, the categorization accuracy is already higher than 99.7%. The

categorization accuracy is not reliable and informative for such highly unbalanced data set. Hence, the submissions were evaluated by using Weighted AUC with the weight distribution shown in Figure 2.9(b). The AUC is only sensitive to the order of the predictions. Instead of predicting true or false, it is more reasonable to submit the probabilities of the true case which is expected to achieve higher score. [27]

The leaderboard on *Kaggle* contains ranking information of all the participants. Before the competition fully ends, the participants could only check the *Public Leaderboard* which reveals the performance of the submission on approximately 40% of the testing data set. On the contrary, the *Private Leaderboard* which discloses the final standings is available after the deadline of the competition. It is based on the remaining 60% of the testing data set. A submission which achieves a high score on the *Public Leaderboard* does not necessarily mean that it could also reach similar score on the *Private Leaderboard*. This mechanism could discourage the participants from overfitting their models to the *Public Leaderboard*. [27]

3.2 Submissions from the participants

The *TUGraz-TUT Face Verification Challenge* was opened for submissions for roughly two months. There were 137 entries submitted by 28 players in 20 teams. The Weighted AUC scores and rankings of the submissions from the participants are shown in Table 3.1. Several benchmark submissions were also included for purposes of comparison. For privacy reasons, the file names had been anonymised except for top 3 performing teams. The Weighted AUC scores in the table were calculated based on the whole testing data set while the corresponding scores on *Kaggle* solely depended on either the public or private testing data set. Consequently, there was tiny discrepancy in the scores. However, the situation was different if the submission which only consists of single value, e.g., one or zero. In order to calculate the Weighted AUC score, one need to plot the ROC curve by varying the discrimination threshold from negative infinity to positive infinity. If the discrimination threshold is less than the single value, all samples are classified as True. Both TPR and FPR are 1. If the discrimination threshold is more than the single value, all samples are classified as False. Both TPR and FPR are 0. In summary, the ROC curve is composed of only two points, namely, $(0, 0)$ and $(1, 1)$. Suppose linear interpolation is applied to fill in the missing values, the interpolated ROC curve is a diagonal line connecting $(0, 0)$ and $(1, 1)$. With the weight distribution shown in Figure 2.9(b), the Weighted AUC score is 0.7. Due to the different implementations of the Weighted

AUC algorithm, the score calculated by *Kaggle* is 0.5. Apart from such special cases, the inconsistency in the scores could be ignored.

It is apparent that the scores of the top 3 performing teams greatly exceed the benchmark solutions which utilizes the Euclidean distance between two *AlexNet* feature vectors. Team *nobody* and *newbie* is from *Graz University of Technology* and *Tampere University of Technology* respectively. Team *chenriwei* is an external participant who does not belong to the host universities. Figure 3.2 shows the highest private score of each winning team with respect to the elapsed days of the competition. Team *newbie* is one of the earliest teams who submitted a submission. Although the first submission did not attain a satisfactory result, they improved their score progressively. Team *nobody* joined in at the middle of this competition and his first submission surpassed the threshold of 0.98. Later on, team *chenriwei* pushed the highest score above 0.99 via his initial submission. Several days before the deadline, team *nobody* achieved great improvement and became the champion. In the following subsections, only the methods proposed by team *nobody* and *newbie* will be discussed as the solution devised by team *chenriwei* is concealed.

3.2.1 Solution from team *nobody*

By using the ensemble of regression trees which is discussed in Section 2.3.2, team *nobody* constructed a landmark detector on top of the Multi-Task Facial Landmark (MTFL) data set [84]. The landmark detector was competent to estimate the coordinates of five facial landmarks which were annotated in the MTFL data set. In the face alignment process, a suitable similarity transform was appraised. It mapped the actual landmark coordinates to the corresponding average coordinates without obliterating the original shape. Each original image was bended into a normalized shape and the *VGG Face* features were extracted from those aligned images. Team *nobody* reached the score 0.99952 with cosine similarity which measures the distance between two feature vectors. In the internal validation data set, the TPR was about 0.77 when the FPR was 0.01. [27]

So as to further improve the results, team *nobody* constructed a 1024-dimensional Large Margin Nearest Neighbors (LMNN) embedding on top of the deep features which had been normalized by Euclidean length. This novel solution improved the score to 0.99969. In the internal validation data set, the TPR was about 0.81 when the FPR was 0.01. [27]

Table 3.1 *The Weighted AUC scores and rankings of the submissions from the participants.*

Team Name	Weighted AUC	Rank	Team Name	Weighted AUC	Rank
nobody	0.9990	01	Anonymous_13	0.8954	13
chenriwei	0.9956	02	Anonymous_12	0.8932	14
newbie	0.9636	03	Anonymous_17	0.8911	15
Anonymous_10	0.9447	04	Anonymous_11	0.8911	16
Anonymous_16	0.9428	05	Benchmark_AlexNet	0.8911	17
Anonymous_05	0.9327	06	Anonymous_15	0.8911	18
Anonymous_04	0.9208	07	Anonymous_01	0.8825	19
Anonymous_08	0.9131	08	Anonymous_07	0.7487	20
Anonymous_14	0.9098	09	Benchmark_all_ones	0.7000	21
Anonymous_06	0.9066	10	Benchmark_all_zeros	0.7000	22
Anonymous_02	0.9002	11	Anonymous_09	0.6962	23
Anonymous_03	0.8970	12	Benchmark_random	0.6883	24

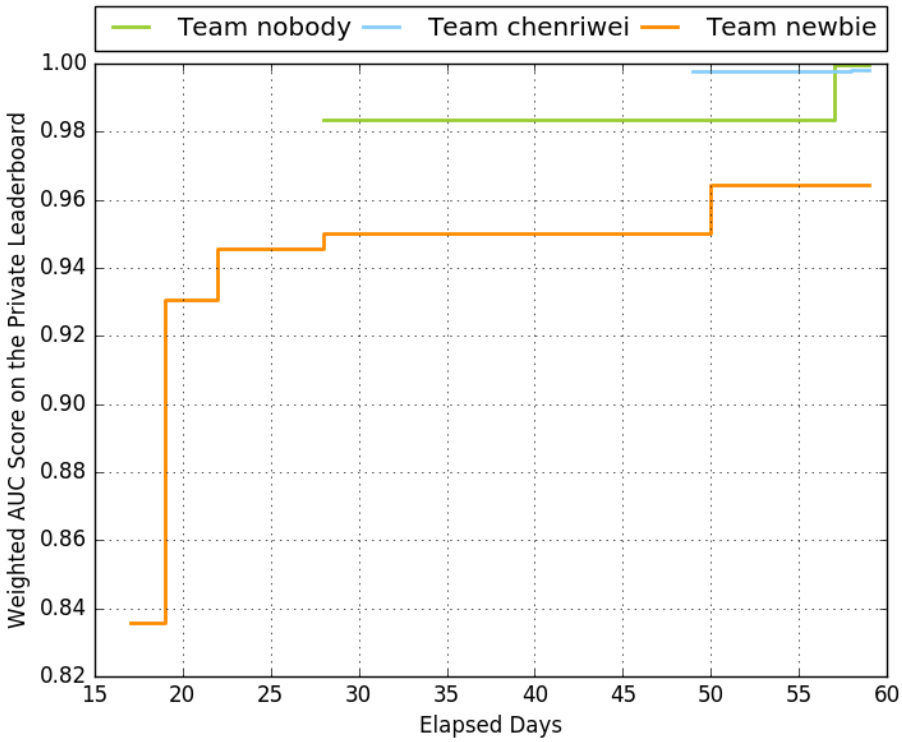


Figure 3.2 *The highest private score of each winning team with respect to the elapsed days.*

3.2.2 Solution from team *newbie*

Team *newbie* exploited the *AlexNet* features. So as to accelerate computation speed, Principal Component Analysis (PCA) was applied to reduce the dimensionality of features from 4096 to 1015 while 99% of the original information was still preserved. [27]

Team *newbie* did not use whether two images represent the same person as the label information. Instead, they treated the parent folder information in the training data set as the labels. A set of binary Support Vector Machine (SVM) classifiers were constructed with one-versus-all strategy. Since the training data set contains images from 271 persons, those SVM classifiers yielded a 271-dimensional vector which represented the probability that the input image belongs to the corresponding person in the training data set. By using the dot product of two 271-dimensional vectors as the final prediction value, team *newbie* achieved the score 0.94480. [27]

Later on, team *newbie* replaced SVM classifiers with Multilayer Perceptron (MLP) which is a feed-forward artificial neural network. The defined MLP contained only one hidden layer. After the coefficients had been adapted to the training data set, a 271-dimensional vector was extracted from the last layer without an activation function. Team *newbie* transformed the Euclidean distance between two 271-dimensional vectors in the same way as the sample solution. After fine-tuning the parameters of the MLP, team *newbie* finally obtained the score 0.96431. [27]

3.3 General analysis

3.3.1 Procedure overview

A typical procedure for this Face Verification task is shown in Figure 3.3. In the cross-validation process, the original training image data set could be split into the training image data set and validation image data set. The training image data set is used to train the machine learning models while the validation image data set is used to evaluate the performance of the models. By selecting two different images from the same data set, one can generate all possible image pairs. Observe that the classification categories are heavily imbalanced, preprocessing is needed before passing the features and labels to the models. The number of False cases will be much

higher than the number of True cases. A straight forward approach is performing random sampling on the majority cases. In the ideal case, the number of occurrences of different classes should be identical. It is noteworthy that such sampling techniques should only be applied to the training data set since the performance of the solution will be evaluated based on all pairs from the testing image data set. Meanwhile, the error metric used in the cross-validation process should comply with the settings of the competition. Otherwise, there could be huge discrepancy between the score calculated by the participants and the score on the leaderboard. After the sampling procedure, the model is trained by using the balanced training data set. Based on the score of the models on the validation data set, one can further fine-tune the parameters of the models and choose the optimal one to generate final predictions for the testing data set.

3.3.2 Generation of features and labels

Figure 3.4 illustrates the process of generation of features. Regardless of what kind of methods the participant intend to use, a sequence of numbers need to be extracted from each image. Such feature vector is a compact representation of the image data. For an image pair chosen from the image data set, two feature vectors are obtained. In the process of difference measure, one may use single difference vector to describe the differences between two images by integrating two feature vectors. Two strategies are appropriate for this purpose. On the one hand, one can simply use the subtraction between two feature vectors in an element-wise manner. It stands to reason that the difference between image 1 and image 2 should be equal to the difference between image 2 and image 1. Therefore, taking the absolute value of the subtraction is more reasonable. On the other hand, there are dozens of predefined distance metrics available, such as Cosine distance, Manhattan distance and Euclidean distance (see Table 3.2). For any distance metric, it may take two vectors as the inputs and returns a scalar value as the output. Each distance metric concentrates on one facet of difference measure and one need to implement feature selection with the intention of getting rid of uninformative distance metrics. The difference vector is treated as the final feature which will be passed to the machine learning models.

Figure 3.5 demonstrates how the labels are created. In the training data set, the images in the same sub-folder belonged to the same person. By comparing the directories of the two images, the label information could be obtained.

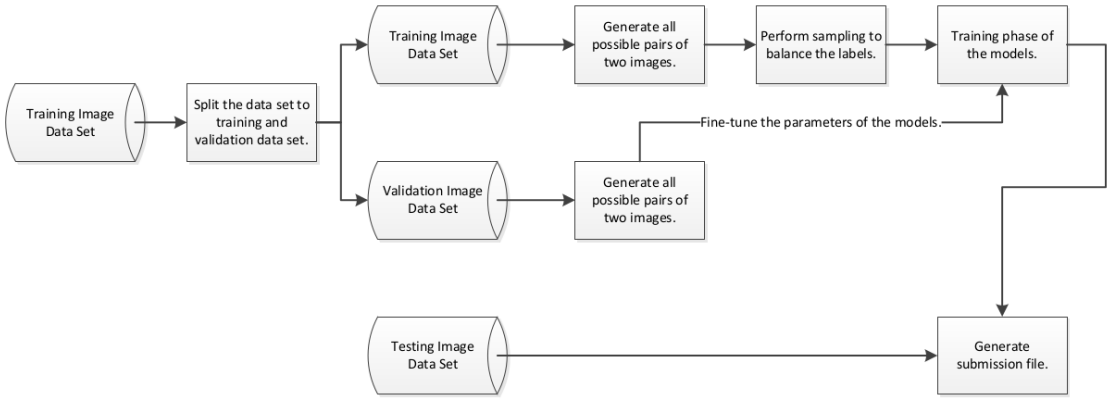


Figure 3.3 *Overview of the procedure.*

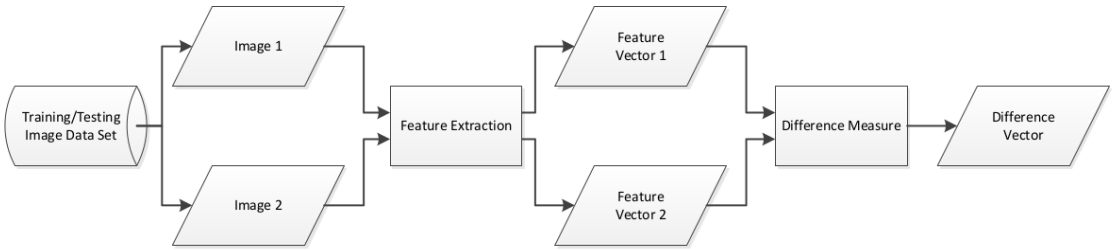


Figure 3.4 *Generation of features.*

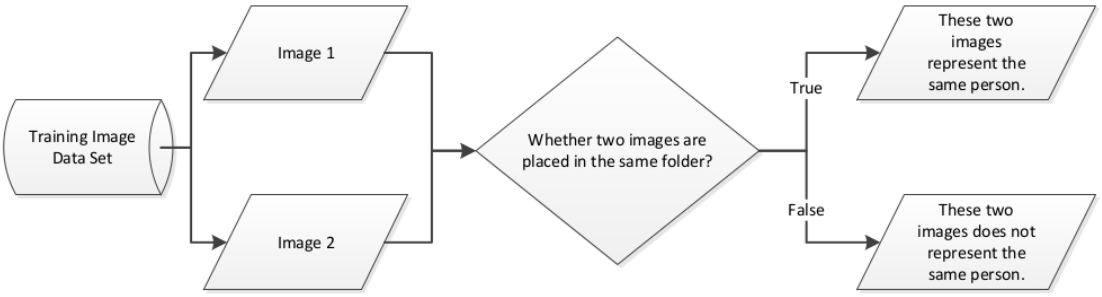


Figure 3.5 *Generation of labels.*

Table 3.2 *A handful of predefined distance metrics [51].*

Name	Definition
Bray-Curtis	$sum(\|x-y\|)/(sum(\|x\|)+sum(\|y\|))$
Canberra	$sum(\|x-y\|/(\|x\|+\|y\|))$
Chebyshev	$sum(max(\|x-y\|))$
Correlation	$cov(x,y)/(std(x)\times std(y))$
Cosine	$sum(x\times y)/(sqrt(sum(x^2))\times sqrt(sum(y^2)))$
$L_1(Manhattan)$	$sum(\|x-y\|)$
$L_2(Euclidean)$	$sqrt(sum((x-y)^2))$
Minkowski	$(sum(\|x-y\|^p))^{1/p}$
Squared Euclidean	$sum((x-y)^2)$

3.3.3 Appropriate cross-validation strategy

As mentioned in Section 3.1, the images from the same person were placed in the same sub-folder. Figure 3.6 illustrates the histogram of number of images from the same person. It is noticeable that the vast majority of the number of images from the same person is less than 8. Since the number of images from the same person could be regarded as the arbitrary domain specific stratifications of the images, there are mainly two strategies to split the original training data to CV folds. On the one hand, one may choose k-fold cross-validation. Assume there are 5 images for a person in total, 3 of them might be assigned to the training data set while the remaining 2 are put in the validation data set. On the other hand, one may select Leave-p-Label-Out cross-validation. The images from the same person will be treated as a whole and all of them will be allocated to either the training data set or the validation data set. In general, Leave-p-Label-Out cross-validation is more appropriate than k-fold cross-validation in this competition and these two approaches is compared in more detail in the following paragraphs.

As a regular rule, the performance of the machine learning model should be evaluated by using the unseen data set. Accordingly, the observations in the training data set must be excluded both from the validation and testing data set. It is possible that the machine learning model may only be capable of distinguishing persons which it has been trained on. In k-fold cross-validation, the images from the same person may exist both in the training and validation data set. As a result, the error metric may give over-optimistic score on the validation data set. Furthermore, the misleading score yielded by the error metric ultimately results in suboptimal solutions and the selected hyperparameters will perform badly on the testing data set. By contrast,

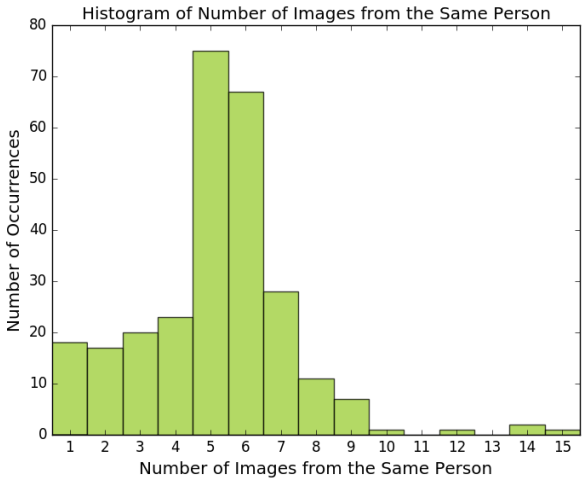
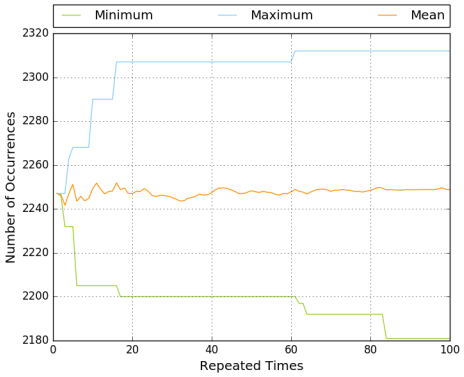
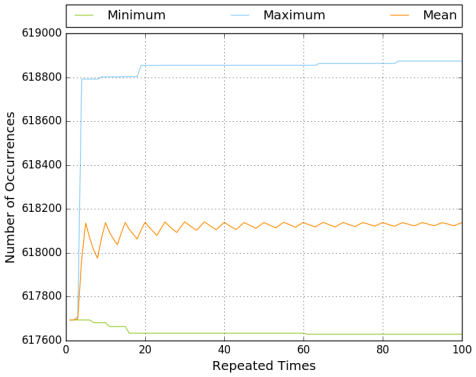


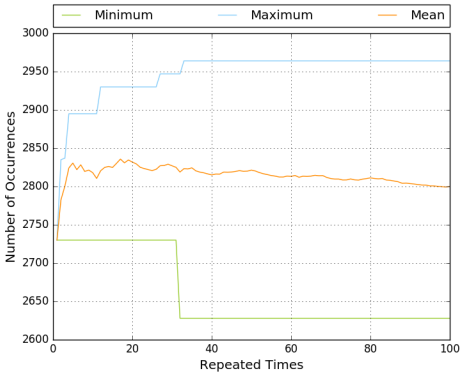
Figure 3.6 *Histogram of number of images from the same person.*



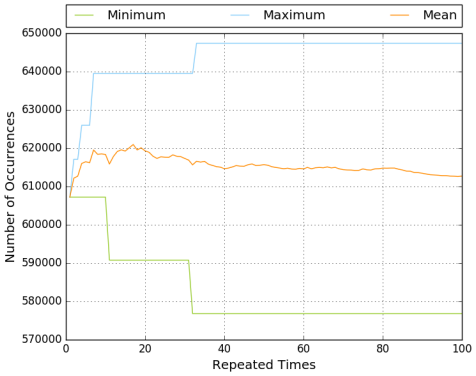
(a) True cases for k-fold cross-validation.



(b) False cases for k-fold cross-validation.



(c) True cases for Leave-p-Label-Out cross-validation.



(d) False cases for Leave-p-Label-Out cross-validation.

Figure 3.7 *Comparison between two cross-validation strategies.*

this issue does not exist in Leave-p-Label-Out cross-validation since all the images from the same person will be treated as a whole rather than individually.

In addition, the number of occurrences of True and False cases are quite different in these two approaches. Although the minimum, maximum and mean value of the number of occurrences of the True or False cases could be retrieved by using mathematical derivation from the histogram shown in Figure 3.6, it is much more straightforward to apply computer simulation which imitates the random splitting process for adequately many times. Considering a 5-fold cross-validation process, 80% images (k-fold cross-validation) or folders (Leave-p-Label-Out cross-validation) will be selected to the training data set while the remaining 20% images or folders will work as the validation data set. Figure 3.7 exemplifies the simulation results of these two cross-validation approaches. The repeated times denotes the number of simulations that has been carried out. For the minimum and maximum value, there is no significant change after the repeated times reaches 50. The mean value is also stabilized as the repeated times increases. Besides, the labels of the vast majority of the pairs are False as the number of occurrences of the True cases is much smaller than that of the False cases. However, the minimum value of the True cases in the latter approach is even higher than the maximum value of the True cases in the former approach. It is an undoubted fact that the mean value of the True cases in the latter approach is noticeably higher than that in the former approach. Even though the disparity in the minimum and maximum value of the False cases in the latter approach are drastically larger, the mean value of the False cases in these two methods are comparatively close to each other. Since the True cases are more valuable in this competition, Leave-p-Label-Out cross-validation is more applicable.

3.4 Analysis of face alignment

3.4.1 Unsupervised joint alignment

Unsupervised Joint Alignment performs rectification of a set of objects and the aligned objects have the same canonical pose. The algorithm is not constrained to faces and it could also work on other objects classes, such as cars. [25] Since the registration process reduces the variability of the objects, it is worthwhile to integrate *Unsupervised Joint Alignment* to solve the face verification challenge.

Figure 3.8 shows three positive examples of *Unsupervised Joint Alignment*. There is

(a) Facial image (*train_ 00000265.jpg*).

(b) Aligned image (*train_ 00000265.jpg*).

(c) Facial image (*train_ 00000262.jpg*).

(d) Aligned image (*train_ 00000262.jpg*).

(e) Facial image (*train_ 00000249.jpg*).

(f) Aligned image (*train_ 00000249.jpg*).

Figure 3.8 *Positive examples of Unsupervised Joint Alignment.*

(a) Facial image (*train_ 00000341.jpg*).

(b) Aligned image (*train_ 00000341.jpg*).

(c) Facial image (*train_ 00000326.jpg*).

(d) Aligned image (*train_ 00000326.jpg*).

(e) Facial image (*train_ 00000147.jpg*).

(f) Aligned image (*train_ 00000147.jpg*).

Figure 3.9 *Negative examples of Unsupervised Joint Alignment.*

a certain level of tilt in the original facial images. The tilt is correctly compensated by rotating the facial image to the opposite direction. Due to the limitation of the implementation of the algorithm, unexpected black areas appear at the edges of the aligned image. Currently, there is no quick fix to suppress this side effect.

Figure 3.9 shows three negative examples of *Unsupervised Joint Alignment*. In Figure 3.9(a), the roll angle of the head is not perceptible. However, the algorithm rotates the facial image counterclockwise by approximately 30 degrees which only makes the aligned image worse than the original one. In Figure 3.9(c), the original facial image leans a little to the right. The algorithm rotates the facial image clockwise which is apparently a mistake. In Figure 3.9(e), the original facial image should be rotated from left to right. The angle of rotation is so small that the facial image is not aligned good enough.

3.4.2 Ensemble of regression trees

Ensemble of Regression Trees is utilized to estimate the coordinates of facial landmarks. In addition, each face is manipulated to make the critical landmarks, i.e., eyes and bottom lip, appear in the similar position. Such manipulation is accomplished by applying an appropriate affine transformation which involves a remapping routine. [2, 29]

Figure 3.10 shows three positive examples of *Ensemble of Regression Trees*. Although the frontal face is visible, the roll angle of the head could not be ignored. In the aligned images processed by the algorithm, the defect of the roll angle has been eliminated. The coordinates of the two eyes have been transformed to the same horizontal level. Such aligned images are ideal for the following processes.

Figure 3.11 shows three negative examples of *Ensemble of Regression Trees*. In addition to the roll angle of the head, the yaw and pitch angle of the head are also noticeable. In spite of the two eyes have been placed on the fixed coordinates, the original facial image has been twisted too much, with the result that the aligned images look to be unnatural for a human. This could result in extra challenges for the following processes.

Figure 3.12 shows five unsuccessful examples of *Ensemble of Regression Trees*. The training data set contains 1393 images in total and the algorithm could not generate

(a) Facial image (*train_ 00000405.jpg*).

(b) Aligned image (*train_ 00000405.jpg*).

(c) Facial image (*train_ 00000163.jpg*).

(d) Aligned image (*train_ 00000163.jpg*).

(e) Facial image (*train_ 00000093.jpg*).

(f) Aligned image (*train_ 00000093.jpg*).

Figure 3.10 *Positive examples of Ensemble of Regression Trees.*

(a) Facial image (*train_ 00000289.jpg*).

(b) Aligned image (*train_ 00000289.jpg*).

(c) Facial image (*train_ 00000649.jpg*).

(d) Aligned image (*train_ 00000649.jpg*).

(e) Facial image (*train_ 00001027.jpg*).

(f) Aligned image (*train_ 00001027.jpg*).

Figure 3.11 *Negative examples of Ensemble of Regression Trees.*

(a) Original image (*train_ 00000421.jpg*).

(b) Original image (*train_ 00001042.jpg*).

(c) Original image (*train_ 00001107.jpg*).

(d) Original image (*train_ 00000677.jpg*).

(e) Original image (*train_ 00001160.jpg*).

Figure 3.12 *Unsuccessful examples of Ensemble of Regression Trees.*

Figure 3.13 *Mean of the original facial images.*

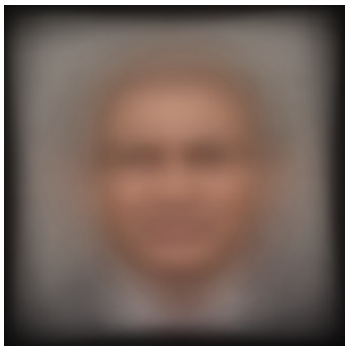


Figure 3.14 *Mean of the aligned images processed by Unsupervised Joint Alignment.*

Figure 3.15 *Mean of the aligned images processed by Ensemble of Regression Trees.*

the aligned image for 86 images among them. In those five examples, only one side of the face is visible while the other half is totally missing. Without estimating the other half of the face, the affine transformation algorithm does not work in such extreme cases.

3.4.3 Perceptual comparison of face alignment algorithms

Figure 3.13 illustrates the mean of the original facial images and it is quite blurry. Figure 3.14 and Figure 3.15 demonstrate the mean of the aligned images processed by *Unsupervised Joint Alignment* and *Ensemble of Regression Trees*, respectively. In Figure 3.14, the black areas could be found around the borders. In Figure 3.15, the eyes and nose are much more clear than the other two images. Moreover, the forehead is omitted during the face alignment process.

Ensemble of Regression Trees can suppress not only the roll angle but also the yaw and pitch angles while *Unsupervised Joint Alignment* works merely on fixing the deviation of the roll angle. Moreover, the black areas generated by *Unsupervised Joint Alignment* are also undesirable. In general, one may have an initial assumption that *Ensemble of Regression Trees* works better than *Unsupervised Joint Alignment*.

3.5 Analysis of difference measure

scikit-learn natively supports a variety of distance metrics (see Table 3.3) [51]. Instead of taking the absolute value of the subtraction between two feature vectors,

the top 10 important distance metrics are selected and concatenated as the final feature vector which describes the difference between two faces. This could reduce the dimensionality of the final features greatly and also decrease the computational complexity of the final classifier. With the purpose of assessing the importance of each distance metric, a random forest classifier is trained by feeding the distance values. The importance of each distance metric could be retrieved from such classifier and a dependable result could be further obtained by repeating this process multiple times and taking the average value.

Table 3.3 *Distance metrics natively supported by scikit-learn [51].*

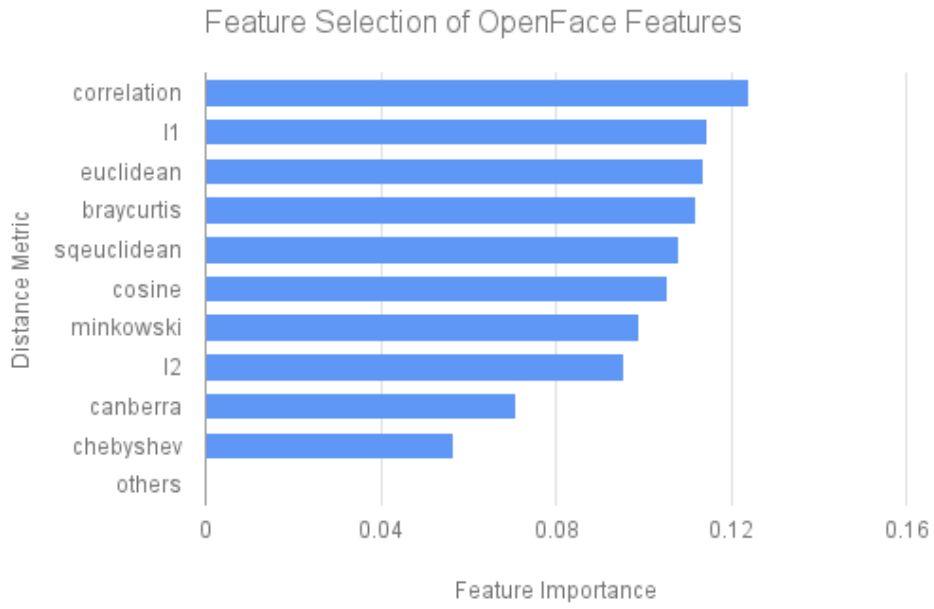
braycurtis	canberra	chebyshev	correlation	cosine
dice	euclidean	hamming	jaccard	kulsinski
l1	l2	matching	minkowski	rogerstanimoto
russellrao	sokalmichener	sokalsneath	sqeuclidean	

3.5.1 Feature selection

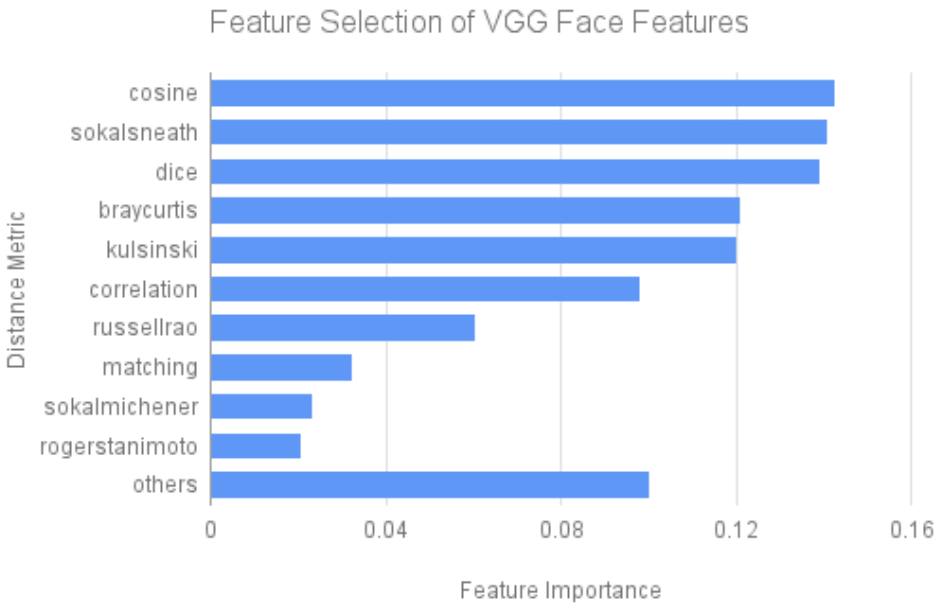
Figure 3.16 compares the result of the feature selection of the *OpenFace* and *VGG Face* features. The item *others* refers to the cumulative feature importance of the remaining distance metrics which are not the top 10 important distance metrics. Figure 3.16(a) gives information on the importance of distance metrics for the *OpenFace* features which are retrieved on top of the aligned images processed by *Ensemble of Regression Trees*. The *correlation* and *l1* are the top 2 distance metrics. Figure 3.16(b) presents information about the importance of distance metrics for the *VGG Face* features which are calculated on top of the original facial images. The *cosine* and *sokalsneath* are the top 2 distance metrics. For the *OpenFace* features, the *others* could be omitted since the cumulative sum is very close to 0. However, the importance of the *others* still takes up approximately 0.10 for the *VGG Face* features. Since only the top 10 distance metrics are chosen, the overall performance of the *VGG Face* features might decline a little bit due to the information loss.

3.5.2 Visualization of top 2 distance metrics

Figure 3.17 visualizes the top 2 distance metrics of the *OpenFace* and *VGG Face* features, respectively. Sampling has been used to balance the occurrence of True

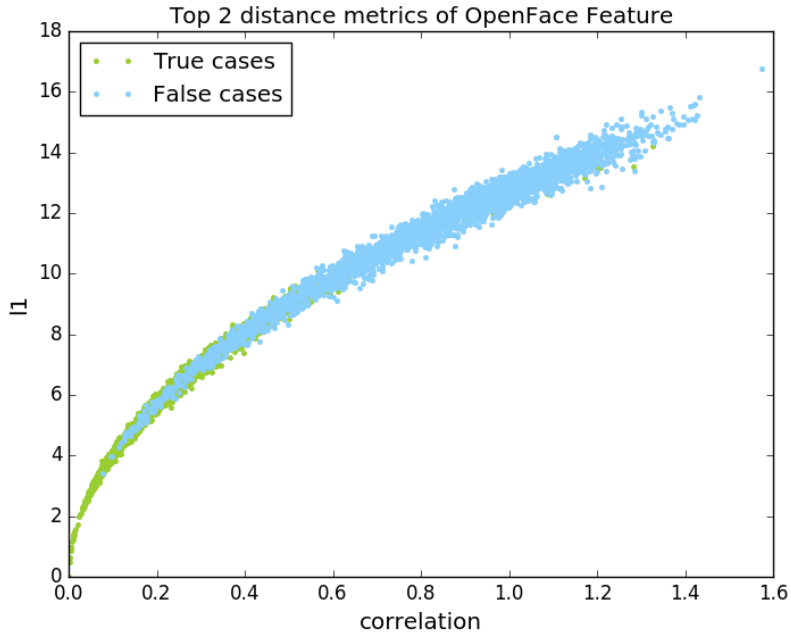


(a) Feature selection of *OpenFace* features with the aligned images processed by *Ensemble of Regression Trees*.

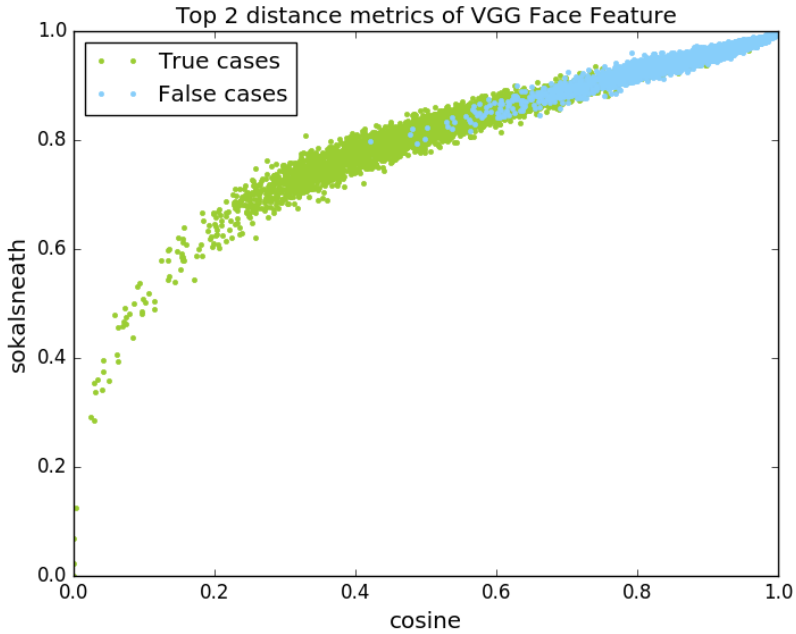


(b) Feature selection of *VGG Face* features with the original facial images.

Figure 3.16 *Feature selection.*



(a) Top 2 distance metrics of *OpenFace* features with the aligned images processed by *Ensemble of Regression Trees*.



(b) Top 2 distance metrics of *VGG Face* features with the original facial images.

Figure 3.17 *Visualization of top 2 distance metrics.*

and False cases. In Figure 3.17(a), the True and False cases overlap with each other especially when the value of distance metric is high. In Figure 3.17(b), the intersection of the True and False cases is relatively smaller. The cases which the *cosine* is smaller than 0.5 are highly likely to be True cases. For both the *OpenFace* and *VGG Face* features, the illustrated curve could be treated as a monotonically increasing function generally. If a observation has higher value on one distance metric, it also tends to attain higher value on another distance metric. One should bear in mind that such visualization is not strongly related to final performance of the features. Suppose there is no overlap between the True and False cases at all, one could expect that such feature is a perfect representation. However, the contrary does not hold true. In addition to these top 2 distance metrics, there are other 8 distance metrics which contain extra information. Besides, the feature itself could encompass hidden patterns that are invisible in such two-dimensional figures.

3.6 Analysis of classifiers

In this section, the principle mechanism of two classifiers are enlightened. Cross-validation is integrated in order to obtain the optimal model. After the cross-validation process, one may retrain a new classifier with the optimal hyperparameters on the whole training data set. However, suitable ensemble strategies deem to improve the result slightly. It is possible to use those classifiers which have already been trained during the cross-validation process to generate the predictions for the testing data set. Since the error metric is Weighted AUC, one should use the probabilities rather than the classes. Therefore, using the mean or median value of all the predictions as the final prediction is a wise choice.

3.6.1 Neural network

Figure 3.18 depicts the structure of the Neural Network. The configuration is inspired by an example code shipped along with *Keras* [14]. The NN is relatively small since the dimensionality of the features is only 10. A smaller NN could also help to fight against overfitting as larger NN has higher learning capability which might mislead itself to the noises. The basic element is a fully connected *Dense* layer, a *PReLU Activation* layer, a *BatchNormalization* layer followed by a *Dropout* layer. Such basic element is repeated for three times. At the end of the NN, additional *Dense* and *Activation* layer are appended.

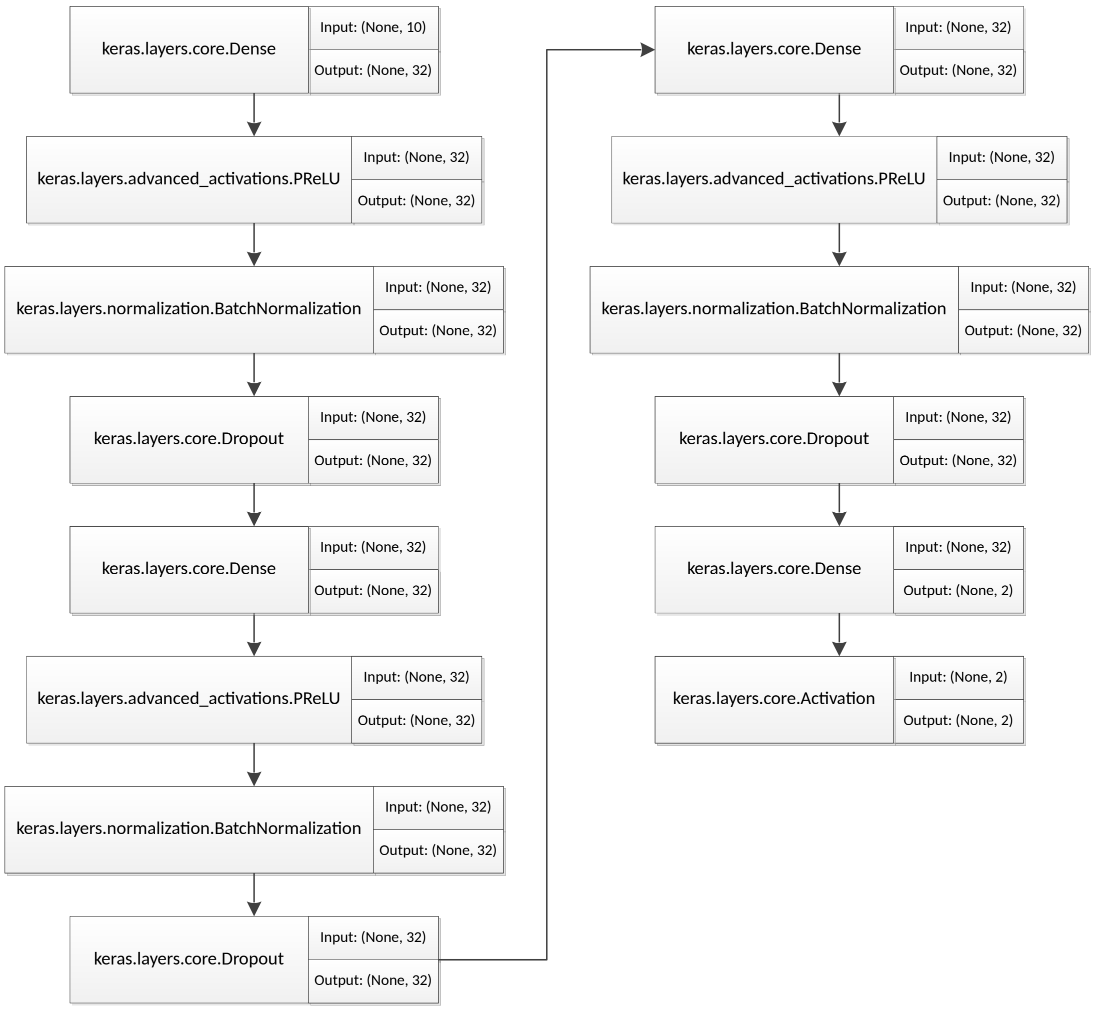


Figure 3.18 *Structure of the Neural Network.*

With the purpose of improving the generalization of the NN, cross-validation is utilized. As discussed in Section 3.3.3, the original training images need to be divided by employing the Leave-p-Label-Out cross-validation strategy. The score on the validation data set has been monitored during the whole training phase. Normally, the validation score improves at the initial phase of training. After the NN has already fit quite well to the training data set, the validation score does not further improve as the training phase still continues. One need to decide when the training phase should be terminated based on the validation score since overfitting is likely to decrease the performance on the unseen data set. Early Stopping is one possible strategy to address this issue. The training will be stopped if there is no improvement in the validation score after a certain number of epochs which could be

named as *patience*. However, it is fiddly to set the value of *patience*. If the *patience* is too small, one might miss the actual optimal epoch. If the *patience* is too high, the NN might have already been polluted by overfitting. A better practise could be setting a high value of the total number of epochs and saving the model after every epoch. The model file will be updated only under the condition that the new model gains a higher validation score. By taking the latter approach, one is guaranteed to obtain the possible optimal model at the expense of running unnecessary epochs.

3.6.2 Support vector classification

Support vector machines (SVMs) are a set of supervised machine learning algorithms. In a basic SVM model, the data set could be treated as points in the high dimensional space. The algorithm searches for a set of hyperplanes which could separate the data set assigned to different categories as much as possible. Among SVMs, support vector classification (SVC) is devised to solve twoclass and multiclass classification problems. [10, 20] Table 3.4 lists three important hyperparameters for a *SVC* classifier, namely, C, kernel and gamma. By exhaustively generating all possible combinations of parameter values shown in the table, one can have 12 *SVC* classifier with different parameter settings. In order to find the optimal parameter setting which could fit the given data set best, the performance of each classifier need to be evaluated during the cross-validation process.

Table 3.4 *Important hyperparameters for a SVC classifier [51].*

Penalty parameter C	Kernel type	Kernel coefficient gamma
[1, 10, 100, 1000]	"linear"	"auto"
[1, 10, 100, 1000]	"rbf"	[0.001, 0.0001]

3.7 Comprehensive analysis of the learning system as a whole

Table 3.5 itemizes the possible options within each phase of the learning system. By selecting one option for each phase, there are 72 different combinations in total. In this section, the influence of each option is scrutinized while the other options are kept constant. By default, the remaining tables in this section will be sorted by the score in descending order.

Table 3.5 *Possible options within each phase of the learning system.*

Phase	Option
Face alignment	Without alignment
	Unsupervised Joint Alignment [25]
	Ensemble of Regression Trees [29]
Feature extraction	*OpenFace* [2]
	VGG Face [50]
Classifier	Neural network
	Support vector classification
Ensemble strategy	Without ensemble
	Mean or median value
Error metric	Weighted AUC
	TPR at a low FPR
	Matthews correlation coefficient

3.7.1 Effects of face alignment and feature extraction

The effects of face alignment and feature extraction are analyzed with classifier, ensemble strategy and error metric set to neural network, mean value and Weighted AUC, respectively. Table 3.6 shows the detailed result of 6 submissions. As the lowest score achieved with the *VGG Face* features is still higher than the highest score achieved with the *OpenFace* features, the *VGG Face* features work better than the *OpenFace* features in general. Additionally, the face alignment algorithm does not make a huge difference with the *VGG Face* features. However, the *OpenFace* features need to be extracted from the aligned images processed by *Ensemble of Regression Trees*. Otherwise, the performance will degrade sharply. The reason is that the deep neural network in *OpenFace* was explicitly trained with the aligned images processed by *Ensemble of Regression Trees*. As discussed in Section 3.4.3, the aligned images processed by different algorithms differ from each other greatly. Consequently, the deep neural network in *OpenFace* generated unproductive feature vectors for the original facial images and the aligned images processed by *Unsupervised Joint Alignment*.

3.7.2 Effects of classifier and ensemble strategy

The effects of classifier and ensemble strategy are analyzed with face alignment, feature extraction and error metric set to *Ensemble of Regression Trees*, *OpenFace* and

Table 3.6 *Effects of face alignment and feature extraction.*

Face alignment	Feature extraction	Weighted AUC
Without alignment	*VGG Face* [50]	0.9983
Unsupervised Joint Alignment [25]	*VGG Face* [50]	0.9979
Ensemble of Regression Trees [29]	*VGG Face* [50]	0.9930
Ensemble of Regression Trees [29]	*OpenFace* [2]	0.9857
Without alignment	*OpenFace* [2]	0.8999
Unsupervised Joint Alignment [25]	*OpenFace* [2]	0.8083

Weighted AUC, respectively. Table 3.7 shows the detailed result of 14 submissions. As discussed in Section 3.3.3, the images in 20% folders work as the validation data set each time and 5 models will trained in one iteration. The score of single model varies from 0.9837 to 0.9857. Although the variance between two single models is quite small, the neural network works better than the SVC classifier as the worst neural network achieves 0.9856 while the best SVC classifier achieves 0.9846. It is also noticeable that taking mean or median value of the predictions generated by neural networks or SVC classifiers could not improve the Weighted AUC score in this competition. Cosine distances between the predictions generated by single models of neural networks or SVC classifiers are calculated separately (see Table 3.8 and Table 3.9). The average value of the non-zero elements in the preceding tables are 0.00191 and 0.00280, respectively. One could not benefit from the ensemble strategies since the discrepancies within the predictions generated by single models could be neglected.

3.7.3 Effects of error metric

In this subsection, the effects of error metric will be analyzed. Moreover, the submissions from the top 3 performing teams in the competition are also included for comparison. In order to simplify the evaluation process, only the optimal submissions with either *OpenFace* or *VGG Face* features are taken into consideration. Table 3.10 shows the rankings of the solutions with different error metrics. Generally speaking, the rankings are consistent with each other. The only exception is the rank of team *chenriwei* with the error metric MCC. On the one hand, the Weighted AUC score of team *nobody*, *VGG Face* and team *chenriwei* are above 0.995. The difference in the Weighted AUC scores of any two submissions is quite small. On the other hand, the scores vary a lot with the other two error metrics. In summary,

Table 3.7 *Effects of classifier and ensemble strategy.*

Classifier	Ensemble strategy	Weighted AUC
Neural network	Without ensemble (*Model 2*)	0.9857
Neural network	Without ensemble (*Model 5*)	0.9857
Neural network	Without ensemble (*Model 1*)	0.9857
Neural network	Mean value	0.9857
Neural network	Median value	0.9857
Neural network	Without ensemble (*Model 4*)	0.9857
Neural network	Without ensemble (*Model 3*)	0.9856
Support vector classification	Without ensemble (*Model 1*)	0.9846
Support vector classification	Without ensemble (*Model 3*)	0.9844
Support vector classification	Mean value	0.9843
Support vector classification	Median value	0.9843
Support vector classification	Without ensemble (*Model 4*)	0.9841
Support vector classification	Without ensemble (*Model 2*)	0.9837
Support vector classification	Without ensemble (*Model 5*)	0.9837

Table 3.8 *Cosine distances between the predictions generated by neural networks.*

	Model 1	*Model 2*	*Model 3*	*Model 4*	*Model 5*
Model 1	0	0.00220	0.00034	0.00061	0.00158
Model 2	0.00220	0	0.00283	0.00202	0.00378
Model 3	0.00034	0.00283	0	0.00047	0.00216
Model 4	0.00061	0.00202	0.00047	0	0.00304
Model 5	0.00158	0.00378	0.00216	0.00304	0

Table 3.9 *Cosine distances between the predictions generated by SVC classifiers.*

	Model 1	*Model 2*	*Model 3*	*Model 4*	*Model 5*
Model 1	0	0.00290	0.00426	0.00018	0.00074
Model 2	0.00290	0	0.00108	0.00325	0.00481
Model 3	0.00426	0.00108	0	0.00450	0.00584
Model 4	0.00018	0.00325	0.00450	0	0.00047
Model 5	0.00074	0.00481	0.00584	0.00047	0

TPR at a low FPR and MCC are more informative than Weighted AUC in this competition.

Table 3.10 *The rankings of the solutions.*

Solution	Weighted AUC	Rank	TPR at a low FPR (0.01)	Rank	MCC	Rank
nobody	0.9990	1	0.8607	1	0.6417	1
VGG Face	0.9990	2	0.7609	2	0.5449	3
chenriwei	0.9956	3	0.6740	3	0.5474	2
OpenFace	0.9857	4	0.3363	4	0.1852	4
newbie	0.9636	5	0.2301	5	0.1331	5

3.7.4 Comparison of the computational speed

In pursuance of comparing the computational speed, the program was executed by using a laptop with the configuration shown in Table 3.11. GPU acceleration was activated whenever possible. The execution time was mainly constrained by CPU and GPU since the test scenarios could not saturate the hard disk and memory. By repeating a specific operation for a large number of times, one could get a dependable result.

Table 3.12 compares the computational speed of the algorithms. In the process of face alignment, it takes much less time for the *Ensemble of Regression Trees* algorithm to align one image. Since the aligned images processed by *Unsupervised Joint Alignment* could not improve the score of the learning system, this algorithm is obsolete. In the process of feature extraction, *OpenFace* is approximately 16 times faster than *VGG Face* as the size of the deep neural network in *OpenFace* is much smaller than that in *VGG Face*. In the process of training and testing phase of the classifier, it is apparent that training one model could be accomplished within minutes. Meanwhile, generating a prediction for one image pair takes less than 0.002 second. Although the classifiers work faster with the *OpenFace* features in the testing phase, the advantage is negligible.

Table 3.11 *Computer configuration.*

CPU	Intel Core i5-3230M @ 2.60GHz
GPU	NVIDIA GeForce GT 740M 2GB DDR3
Hard Disk	Samsung SSD 850 EVO 1 TB
Memory	12 GB 1600 MHz DDR3 SODIMM
Operating System	Arch Linux x86_64 with the bleeding-edge packages until February 2016

Table 3.12 *Comparison of the computational speed.*

Face alignment		
Measurement	*Unsupervised Joint Alignment* [25]	*Ensemble of Regression Trees* [29]
Process one image	1.1112 second	0.8250 second
Feature extraction		
Measurement	*OpenFace* [2]	*VGG Face* [50]
Process one image	0.0146 second	0.2434 second
Classifier with the *OpenFace* features [2]		
Measurement	Neural network	Support vector classification
Train one model	68.0 second	23.5 second
Predict one pair	0.0014 second	0.0013 second
Classifier with the *VGG Face* features [50]		
Measurement	Neural network	Support vector classification
Train one model	84.4 second	12.4 second
Predict one pair	0.0019 second	0.0018 second

4. CONCLUSIONS

In this thesis, the solutions proposed by the top performing teams in *TUGraz-TUT Face Verification Challenge* along with other possible methods were summarized. By leaning on the *OpenFace* and *VGG Face* features, those exceptional approaches obtained good results in face verification. The Chinese Realist Confucian philosopher Xunzi states that "Not having heard something is not as good as having heard it; having heard it is not as good as having seen it; having seen it is not as good as knowing it; knowing it is not as good as putting it into practice." [80]. Based on the feedback received from students, this competition helped them to apply the knowledge learned from lectures to solve the real-world problems [27].

To further refine the face verification system, several possible areas of further work have been outlined as follows. Firstly, one could collect more facial images which are freely available to the academia. As stated by many researchers, the deep neural network could benefit from larger image data set [2,39,61,70]. The giant companies such as Google and Facebook have inherent advantages in term of the amount of images. However, those companies might be unable to share their proprietary data set due to the constraints of law. Secondly, current face alignment techniques are not perfect since they do not work if only half face is shown. Smearing the available facial image to a 3D model might be one possible approach to address this tricky problem. The missing part of the face could be reconstructed afterwards. Thirdly, one could investigate the reliability of the error metrics in detail. The Weighted AUC could not differentiate the proposed solutions very well. Fourthly, one could implement a real-time face verification system with the *OpenFace* features. Computational optimization is still required especially for the face alignment algorithm. Finally, one could replace the classifier with the *eXtreme Gradient Boosting* [11]. This algorithm gains a lot of popularity among the Machine Learning enthusiasts since it has been applied to win ten *Kaggle* competitions so far [12].

BIBLIOGRAPHY

[1] C. C. Aggarwal, *Data classification: algorithms and applications*. CRC Press, 2014.

[2] B. Amos, B. Ludwiczuk, J. Harkes, P. Pillai, K. Elgazzar, and M. Satyanarayanan, "Openface: Face recognition with deep neural networks," 2016. [Online]. Available: http://github.com/cmusatyalab/openface

[3] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[4] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Recognizing facial expression: machine learning and application to spontaneous behavior," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 2005, pp. 568–573.

[5] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[6] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[7] V. Blanz, P. Grother, J. P. Phillips, and T. Vetter, "Face recognition based on frontal views generated from non-frontal images," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, 2005, pp. 454–461.

[8] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[9] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[10] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *arXiv preprint arXiv:1603.02754*, 2016.

[12] T. Chen, T. He, and M. Benesty, "Machine learning challenge winning solutions," 2016. [Online]. Available: https://github.com/dmlc/xgboost/tree/master/demo

[13] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] F. Chollet, "Keras," 2015. [Online]. Available: https://github.com/fchollet/keras

[15] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 539–546.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2009, pp. 248–255.

[17] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[18] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[19] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[20] L. A. Gabralla, R. Jammazi, and A. Abraham, "Oil price prediction using ensemble machine learning," in *Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on*, 2013, pp. 674–679.

[21] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 6645–6649.

[22] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks*, vol. 2, no. 2004, 2004.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[24] ——, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[25] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 2007, pp. 1–8.

[26] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," Tech. Rep., 2007.

[27] K. Inc, "Tugraz-tut face verification challenge," 2015. [Online]. Available: https://inclass.kaggle.com/c/face-verification2

[28] ——, "Kaggle competitions," 2016. [Online]. Available: https://www.kaggle.com/solutions/competitions

[29] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 1867–1874.

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] W. Knight, "Five lessons from alphago's historic victory," 2016.

[32] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2, 1995, pp. 1137–1145.

[33] S. Kotsiantis, D. Kanellopoulos, P. Pintelas *et al.*, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural*

Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[35] E. Learned-Miller, G. Huang, A. RoyChowdhury, H. Li, G. Hua, and G. B. Huang, "Labeled faces in the wild: A survey," 2015.

[36] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[38] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[39] J. Liu, Y. Deng, and C. Huang, "Targeting ultimate accuracy: Face recognition via deep embedding," *arXiv preprint arXiv:1506.07310*, 2015.

[40] D. Los Angeles Thomas *et al.*, *Elementary signal detection theory*. Oxford University Press, 2001.

[41] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[42] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[43] G. Meurant, *Advances in Computers*, ser. Advances in Computers. Elsevier Science, 1993, no. v. 37. [Online]. Available: https://books.google.fi/books?id=vL-bB7GALAwC

[44] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, ser. Symbolic Computation. Springer Berlin Heidelberg, 2013. [Online]. Available: https://books.google.fi/books?id=-eqpCAAAQBAJ

[45] T. Mitchell, *Machine Learning*. McGraw-Hill Boston, MA:, 1997.

[46] ——, "The discipline of machine learning," 2006.

[47] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *Image Processing (ICIP), 2014 IEEE International Conference on*, 2014, pp. 343–347.

[48] G. U. of Technology, "Proclassify user's guide," 2006. [Online]. Available: https://genome.tugraz.at/proclassify/help/pages/XV.html

[49] L. Pappano, "The year of the mooc," *The New York Times*, vol. 2, no. 12, p. 2012, 2012.

[50] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," *Proceedings of the British Machine Vision*, vol. 1, no. 3, p. 6, 2015.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[52] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost, "Machine learning for targeted display advertising: Transfer learning in action," *Machine learning*, vol. 95, no. 1, pp. 103–127, 2014.

[53] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[54] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 873–880.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[56] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. [Online]. Available: https://books.google.fi/books?id=8jZBksh-bUMC

[57] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[58] ——, "Memorial resolution," 1990.

[59] B. Schölkopf and C. J. C. Burges, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[60] N. N. Schraudolph, P. Dayan, and T. J. Sejnowski, "Temporal difference learning of position evaluation in the game of go," *Advances in Neural Information Processing Systems*, p. 817, 1994.

[61] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[62] O. G. Selfridge, "Pandemonium: a paradigm for learning in mechanisation of thought processes," 1958.

[63] E. Shelhamer, J. Donahue, J. Long, Y. Jia, and R. Girshick, "Diy deep learning for vision: a hands-on tutorial with caffe," 2014. [Online]. Available: http://eccv2014.org/program{_}tutorials/

[64] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[65] P. Simon, *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons, 2013, vol. 72.

[66] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Fisher vector faces in the wild." in *BMVC*, vol. 5, no. 6, 2013, p. 11.

[67] D. Stavens and S. Thrun, "A self-supervised terrain roughness estimator for off-road autonomous driving," *arXiv preprint arXiv:1206.6872*, 2012.

[68] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[69] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[70] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[71] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, p. 2, 2012.

[72] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[73] S. University, "Convolutional neural networks for visual recognition," 2016. [Online]. Available: http://cs231n.github.io/

[74] R. J. W. Vernon, C. A. M. Sutherland, A. W. Young, and T. Hartley, "Modeling first impressions from highly variable facial images," *Proceedings of the National Academy of Sciences*, vol. 111, no. 32, pp. E3353–E3361, 2014.

[75] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[76] V. N. Vladimir and V. Vapnik, "The nature of statistical learning theory," 1995.

[77] C. G. Weng and J. Poon, "A new evaluation measure for imbalanced datasets," in *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, 2008, pp. 27–32.

[78] D. E. R. G. E. H. R. J. Williams and G. E. Hinton, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[79] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2011, pp. 529–534.

[80] Xunzi and J. Knoblock, *Xunzi: A Translation and Study of the Complete Works*, 1990. [Online]. Available: https://books.google.fi/books?id=DNqmAAAAIAAJ

[81] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Aggregate channel features for multi-view face detection," in *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, 2014, pp. 1–8.

[82] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[83] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[84] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "Facial landmark detection by deep multi-task learning," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 94–108.