

Abstract

A MobileNet based deep learning model for masked face authentication with computational efficiency

Youngho Son

Computer Engineering

The Graduate School

Sejong University

Biometric technology is being actively used as an identity authentication method. Among biometric technologies, contact methods such as fingerprint recognition have become popular in public institutions and mobile environments. However, contact can cause discomfort and rejection. Currently, in the case of mobile devices, a face recognition system is mainly used, but the recognition rate is low when the face is covered. In addition, due to the characteristics of embedded devices, it is essential to reduce the weight of the model for a small capacity.

In this paper, we propose a face recognition algorithm applied to the applied MobileNet V3 model and the Siamese Network, which reduces the weight of the model through weight sharing. The image pixel array is put into a Siamese Net consisting of two identical networks in a one-shot learning method. The two networks of Siamese Net consist of two networks made shallow in the existing layer of MobileNetV3. The two networks are merged into one at the end and create a filter that can classify faces while sharing weights. The database used to classify images of faces wearing masks is Sejong Face Database. The database includes visible light images, infrared images, and thermal images. Among them, only images wearing masks were separately

classified and set as the Test Data Set. For performance evaluation, the area of AUC and Sensitivity were compared when separately tested on a face wearing a scarf, a face wearing a hat, and a normal face image in addition to the image wearing a mask, and the result of recognizing a face with 96.56% accuracy was confirmed. Furthermore, by applying various optimization functions, it was confirmed whether the functions that can be efficiently learned in the network within a specific period and functions converge stably. The algorithm proposed in this paper is expected to be efficiently utilized in various types of embedded devices.

Keywords: Face recognition, Disguised classification, Lightweight network, Deep learning, MobileNetV3, Siamese Network.

I. Introduction

1. Overview of Biometric Recognition

In modern society, various methods have been introduced to verify one's identity when accessing specific systems. Traditionally, methods requiring physical contact, such as entering passwords and using entry cards, have been primarily used. These methods have been widely applied from accessing company buildings, military facilities, and airports to personal computers and networks. However, these methods required entering a number or carrying a card key each time, regardless of the importance of the information requiring security, which can be inconvenient. Consequently, people had to carry keys and always wear their employee IDs around their necks, resulting in time and economic losses in case of loss. Therefore, simpler methods were required according to the scale and importance of the information.

In contrast to traditional methods, biometric recognition has been presented as an alternative that facilitates easy access authorization and minimizes the risk of information leakage or loss. Biometric recognition extracts physiological and behavioral characteristics to verify identity. By applying this method, it becomes difficult for others to steal passwords or IDs, thus enhancing security. Additionally, it offers convenience to users in situations where information is not highly critical but needs to be accessed frequently. As a result, most companies use biometric recognition methods such as fingerprint recognition, which has also been widely applied to smartphones and laptops.

Biometric recognition can be classified into methods using physiological characteristics such as fingerprints, faces, and irises, and those using behavioral characteristics such as voice, signature, keyboard input patterns, and gait. The classification and characteristics of biometric recognition are shown in Table 1.

Table 1. Classification and Characteristics of Biometric Recognition

Biometric Element	Method	Advantages	Disadvantages
Fingerprint	Contact	Difficult to forge	Fingerprint wear
Face	Contactless	Easy to obtain information	Affected by external factors
Iris	Contactless	Impossible to forge	Recognition angle
Vein	Contact	Easy to obtain information	Relatively high cost
Voice Signal	Contactless	Easy to obtain information	Voice changes
Gait	Contactless	Recognizable from a distance	Low accuracy

The method of identity verification is divided into verification (1:1 matching) and identification (1:N matching). Verification involves comparing input information with stored data to check if they match. Identification involves comparing input information with multiple stored data to find the closest match. This method calculates the similarity between a new image and the N existing images and confirms the identity of the biometric information with the highest similarity.

2. Necessity of This Study

There are various methods to recognize identity using biometric information. Due to the current COVID-19 situation, contact methods may cause discomfort and resistance in gaining user cooperation. People often avoid contact with strangers. Hence, contactless technologies such as facial recognition are increasingly applied in border control systems and airport security. Additionally, unlocking smartphones using facial recognition is already commonly applied in daily life. However, the recognition rate decreases when the face is partially covered, such as when wearing a mask. This issue not only causes inconvenience due to failed recognition but also lowers the reliability of the entire facial recognition system. To address this issue, various methods using machine learning and deep learning have been researched, and this study proposes a deep learning-based facial recognition method as one solution.

Facial recognition research has mainly applied holistic approaches that assume specific distributions of image features to derive low-dimensional representations. Examples include linear subspace, manifold, and sparse representation. These methods suffer from reduced recognition rates in uncontrolled conditions, such as changes in facial angles. These issues are addressed through local feature-based methods. Techniques such as Gabor and LBP play innovative roles in facial recognition and all image processing fields by providing invariance through multi-level and high-dimensional extensions and local filtering. However, these methods have drawbacks such as difficulty in training, slow speed, and vulnerability to lighting changes. Researchers have attempted to improve accuracy through preprocessing, local descriptors, and feature transformation but have not achieved significant breakthroughs.

The advent of deep learning, particularly with AlexNet winning the ImageNet competition in 2012, marked a significant milestone. Deep learning uses multiple layers of processing units to extract and transform features, achieving high invariance to pose, lighting, and emotional changes. While early layers discern human-recognizable features like blue eyes or smiles, later layers create filters for less perceptible features like Gabor and texture. In 2014, DeepFace achieved state-of-the-art results in the LFW benchmark, nearing human performance by introducing the SoftMax Loss function, which proved effective in general object recognition but less so in cases of large intra-variation within the same face. This led to research extending inter-variance through Euclidean-distance-based methods, eventually developing into Contrastive Loss and other margin-based methods.

This study adopts Contrastive Loss and extends the one-shot learning method to few-shot learning using a Siamese network as the base model.

II. Related Work

1. Facial Recognition Using Deep Learning Algorithms

1.1 Siamese Network

A Siamese Network is a model that takes two photos as input, vectorizes both images, and then calculates the Euclidean distance between the two vectors to return their similarity. This model extracts high-quality features that can optimize the given similarity by learning directly from the data rather than using hand-crafted features. As shown in the flowchart in Figure 1, two images are encoded into vectors using shared weights. The similarity between the two vectors is usually defined using the L2 distance in Euclidean space. Through this process, the extracted vectors are trained to have high similarity for similar images and low similarity for different images. The Siamese Network has been used in various tasks such as image recognition, age estimation, gait recognition, object tracking, and sentence similarity. Because it can compare the extracted features of images, it can identify a user's face by comparing features like eye size, shape, and eyebrow shape when the face is covered by a mask.

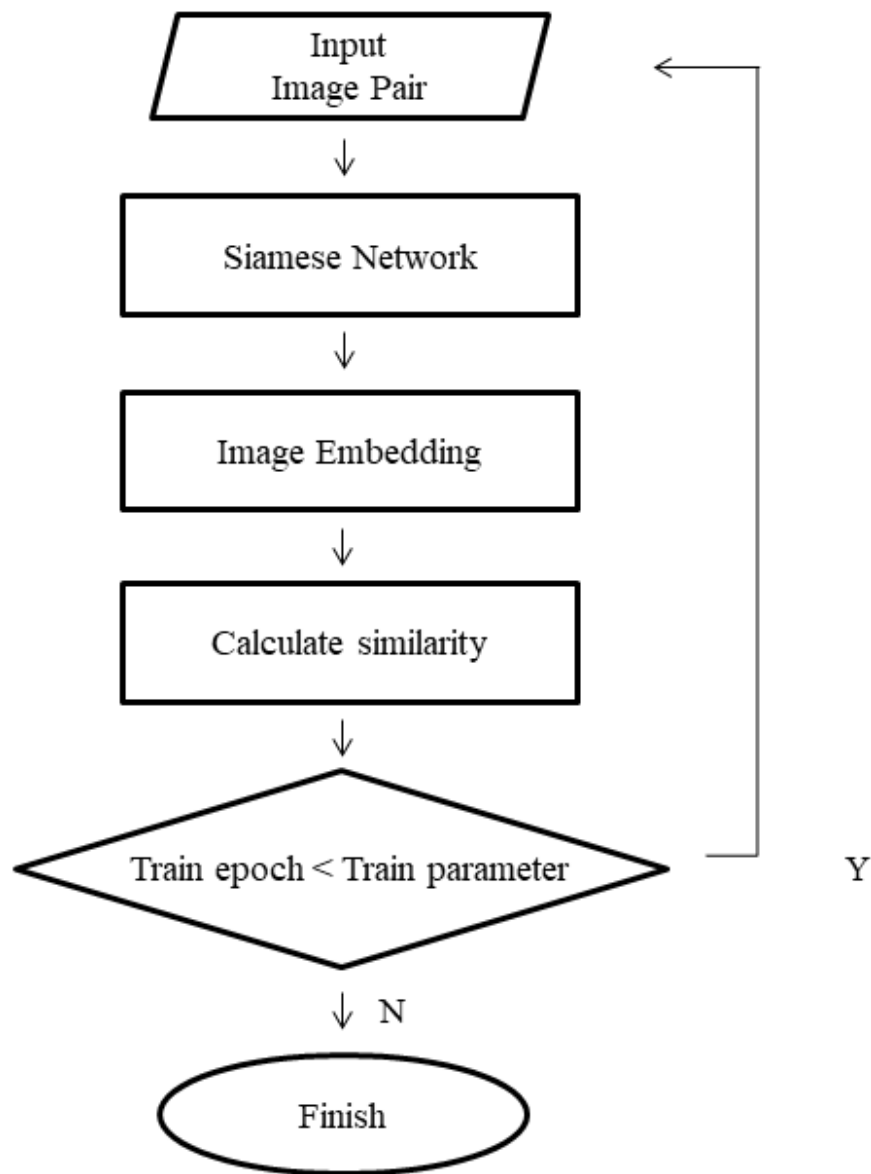


Figure 1. Flowchart of Image Classification Using Siamese Network

1.2 One-shot Learning

When the amount of data is very limited, applying One-shot learning and Few-shot learning methods becomes a good approach. Few-shot learning involves using Support data for training and Query data for validation and is often referred to as the 'N-way K-shot problem', where N is the number of classes, and K is the number of images used for the identification task. For example, in this study, comparing the faces of 63 people in a 1:1 manner would be a One-shot task, while a 1:3 manner in Few-shot learning would be a 63-way 3-shot task. The accuracy tends to improve as the number K in Few-shot learning increases. However, applying Few-shot learning increases the computational load, so the One-shot learning

method was adopted.

In One-shot learning with a small amount of training data, applying loss functions like Cross entropy to train the deep learning model weights can lead to overfitting due to the small number of data points. Therefore, a metric learning approach is needed.

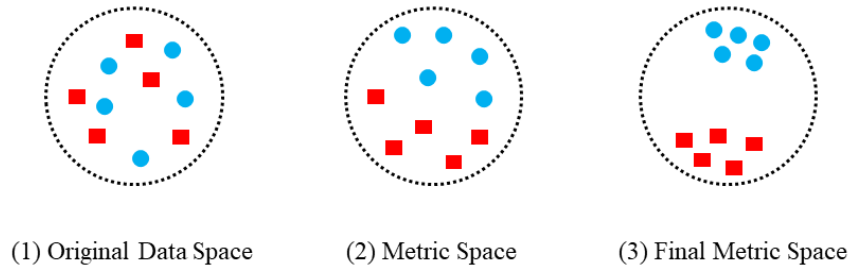


Figure 2. Image of Distance Metric Learning-Based Method

The model, as shown in Figure 2, learns by making the distance between two data points closer if they belong to the same class and farther if they belong to different classes. This study applied Contrastive Loss, a loss function used in metric learning.

$$Loss = Y * D^2 + (1 - Y) * \max(\text{margin} - D, 0)^2 \quad (1)$$

$$Loss_{pred} = Y_{true} * Y_{pred}^2 + (1 - Y_{true}) * \max(\text{margin} - Y_{pred}, 0)^2 \quad (2)$$

When the actual predicted values are substituted into the Contrastive Loss formula (1), it becomes equation (2), where Y is a tensor representing image similarity. Similar images approach 1, while different images approach 0. This is often expressed as Y_{true} . D is a tensor representing the Euclidean distance between image pairs, often expressed as Y_{pred} . Margin is a constant value that can be used to apply the minimum distance between images. If Y is close to 1, the term is given more weight.

1.3 Mobile Net

Just as Generative Adversarial Networks (GANs) consist of two generator and discriminator networks, the Siamese Network is composed of two identical networks. This study used MobileNet, which requires minimal memory, to be used on embedded or mobile devices. MobileNetV1 (MV1) used Depthwise convolution, Pointwise convolution, and Bottleneck structures to achieve efficient computation and preserve image information relative to its size. MobileNetV2 (MV2) improved the V1 Bottleneck structure by introducing an Inverted residual block combined with a Linear Bottleneck for efficient resource utilization. Unlike V1, which maintained or doubled the number of channels, V2 reduced the number of channels using a Projection Layer. MnasNet applied a squeeze and excitation module to the V2 Bottleneck structure, focusing more on the largest representation by placing the Depthwise filter behind the module. MobileNetV3 (V3) was created by combining these techniques. Additionally, V3 applies a method to automatically search and optimize the network. The network uses a Building block combining the Squeeze and excitation module with Bottleneck with residual, as shown in Figure 3.

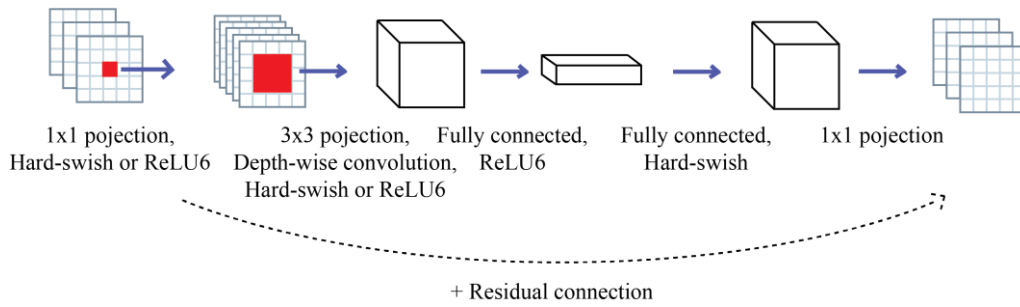


Figure 3. MobileNet V3 Network Block

In equation (3), sigma is the Sigmoid function. The nonlinearity of the Swish function, which replaces ReLU as the activation function, improves accuracy but increases memory operations. Therefore, the ReLU6 function in equation (4) is used, modified to create H-Swish as in equation (5). V3 comes in two versions: Large and Small. This study adopts the Small version, which has a relatively lower accuracy but shallower network depth.

$$swish\ x = x * \sigma(x) \quad (3)$$

$$ReLU = \min(\max(0, x), 6) \quad (4)$$

$$H-swish[x] = x \frac{ReLU(x+3)}{6} \quad (5)$$

2. Optimization Functions

A Siamese Network consists of a function $Gw()$ with shared weights and a Loss module. The input affects the output Loss module of the network. Assuming a pair of images ($X1, X2$) a label Y , the images generate two outputs $G(X1)$ and $G(X2)$ after passing through the network. The Loss module Contrastive Loss generates the distance $Dw(Gw(X1), Gw(X2))$. This Dw and the label Y create a scalar loss value Ls based on the match degree of the labels. The parameter W updates the weights via backpropagation according to the Adaline algorithm. Optimization functions are generated along the weight and loss axes. Perceptrons in deep learning models generate a large number of weights, as shown in Figure 4.

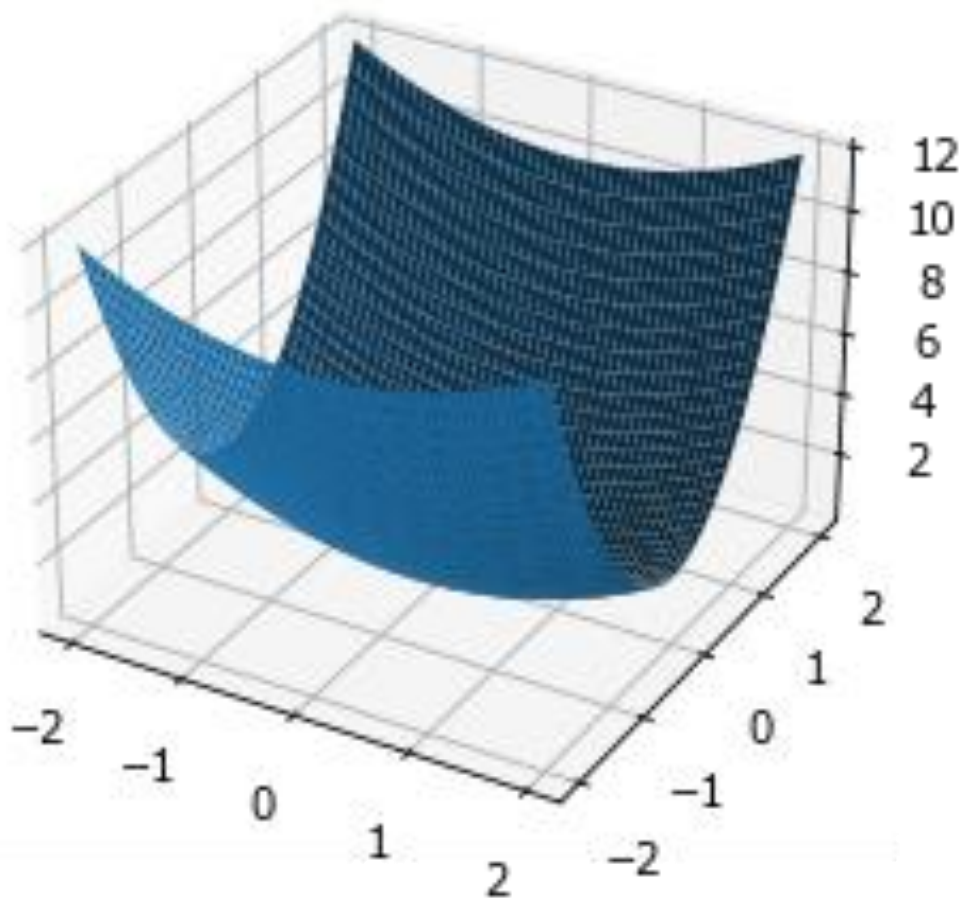


Figure 4. Multivariable Function $x^2 + y^2 = z$

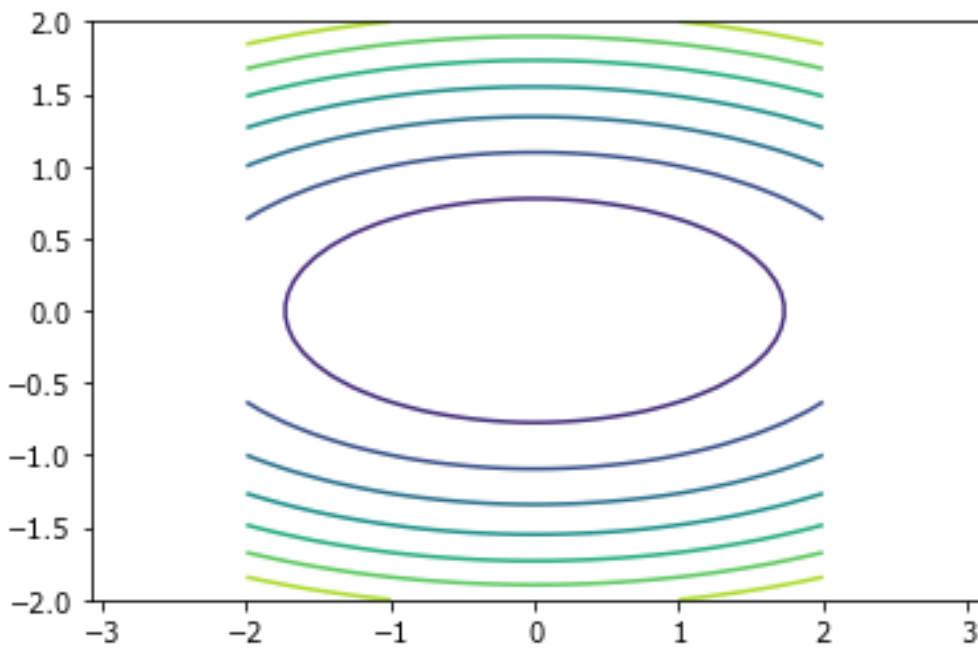


Figure 5. Projection of $x^2 + y^2 = z$

Figure 5 shows the projection of the multivariable function from Figure 4 into two dimensions. The process of adjusting the weights through backpropagation to find the local minima on the function's surface is repeated as shown in Figure 6.

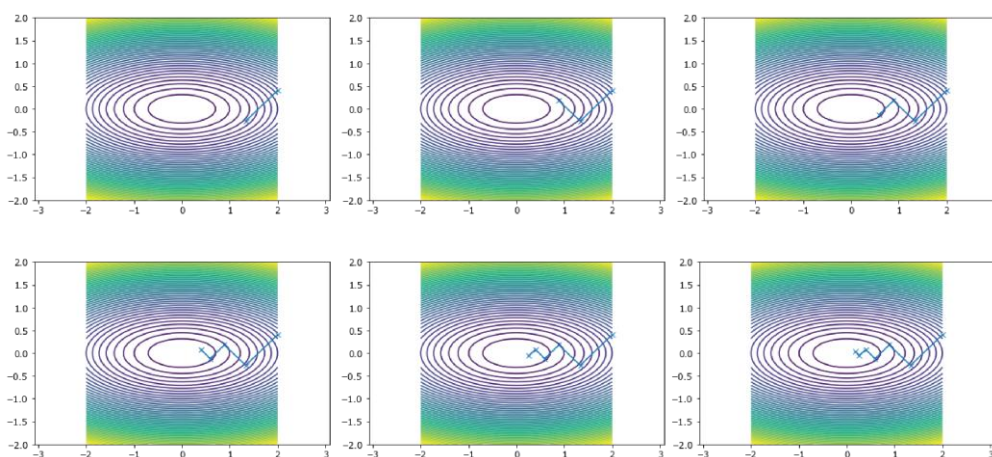


Figure 6. Process of Finding Minima Using Direction Vector in $x^2 + y^2 = z$

There are various optimization techniques, such as Newton method, Gradient descent, and Levenberg-Marquardt (LM). These techniques can be divided into the problem of

determining the direction of the direction vector and the problem of determining the step size for each step. Line search is a method related to the step size. The Gradient Descent method uses the first derivative to find the minima, decreasing if the derivative is positive and increasing if it is negative. Methods such as Newton's method, Gauss-Newton method, and LM apply the second derivative. First-order methods always direct the direction vector toward the minima but have difficulty determining the step size, while second-order methods quickly find the solution using the Taylor series but fail to converge near inflection points and do not distinguish between maxima and minima when determining the direction. Functions like Adagrad, RMSProp, and AdaDelta, which use line search to determine the step size, exist. Momentum-based optimization functions like Momentum and Nesterov-Accelerated Gradient (NAG) focus on determining the direction of the direction vector. These functions introduce the concept of inertia to quickly optimize after calculating the direction vector, as shown in Equation(6).

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right) \quad (6)$$

Equation (7) is a method for optimization using the first derivative, which starts from an arbitrary starting point and varies x until the first derivative (f') converges. Here, is an arbitrary value set by the user.

$$x_{k+1} = x_k - \lambda f'(x_k) \quad (7)$$

Applying this equation to a multivariable function forms the basis of optimization methods like Gradient Descent. Most functions use the Jacobian matrix (equation 8) for optimization, while the Adadelta function uses the Hessian matrix (equation 9).

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \quad (8)$$

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \quad (9)$$

A linear transformation, like shear mapping in Figure 7, moves each point in a fixed direction from coordinates (x, y) to $(x, x+y)$ by an amount proportional to the signed distance on a line parallel to that direction.

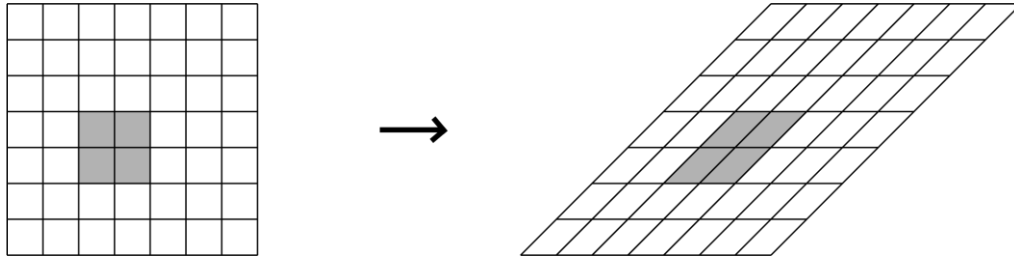


Figure 7. Shear Mapping

The Jacobian matrix is used when attempting a nonlinear transformation that approximates a non-constant coordinate system region to a constant coordinate system in a microscopic region, such as adding $\sin(x)$ as shown in Figure 8. This is because the area of the figure can be approximately measured.

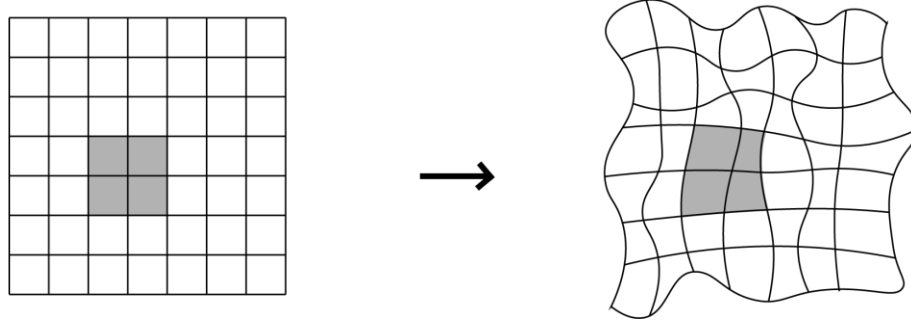


Figure 8. Nonlinear Transformation

The problem with the Jacobian matrix is that it is difficult to determine whether a specific location of a specific function is convex upward, convex downward, or a saddle point at a saddle point. The Hessian matrix can be used to determine that the larger the size of the eigenvalue for a specific eigenvector, the more convex it is by using the eigenvalues and eigenvectors of the second derivative. If all the eigenvalues of the Hessian matrix are positive, the function is convex downward, and if the corresponding part on the coordinates is a critical point, it becomes a local minimum, and if it is positive, it becomes convex upward and becomes a local maximum. If there are negative and positive numbers mixed, it has a saddle shape, and if it is a critical point, it becomes a saddle point. In order to solve the Hessian matrix, the second derivative of the function is calculated, which requires a large amount of computation, so Quasi-Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) that approximately optimize exist, but when applied to experiments, they tend not to converge stably. Therefore, in this study, we applied the Momentum, NAG, Stochastic gradient descent (SGD), RMSProp, and Adadelta optimization functions to verify the stability of accuracy and convergence speed. First, the formula for the SGD optimization function is expressed as Equation (10) if x in Equation (7) is expressed as θ , which means the angle of the weight.:

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} \mathcal{J}(\theta; x^{(i)}, y^{(i)}) \quad (10)$$

$x^{(i)}, y^{(i)}$ indicates that SGD uses one training data set when updating parameters once, and η is used as a correction coefficient.

Compared to Gradient Descent, updates are relatively fast, but the search path may not be efficient for functions whose gradients change depending on the direction, such as anisotropic functions. Momentum is expressed as in Equations (11) and (12).

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta_t} \mathcal{J}(\theta_t) \quad (11)$$

$$\theta_{t+1} = \theta_t - v_t \quad (12)$$

As shown in Figure 6, when the SGD moves in the zigzag direction toward minima, it is reduced by γ times from the previous calculation speed v . Therefore, the direction vector is optimized more quickly as the width of the zigzag toward minima decreases. The disadvantage of using this inertia is that more memory is required because the past speed must be stored. NAG can be expressed as Equations (13) and (14).

$$v_t = \gamma v_{t-1} + \eta \nabla_{\theta_t} \mathcal{J}(\theta_t - \gamma v_{t-1}) \quad (13)$$

$$\theta_{t+1} = \theta_t - v_t \quad (14)$$

Similar to the Momentum method, NAG subtracts speed from a weight called weight or gradient. Looking at the moment update equation, it becomes $\theta_{t+1} = \theta_t - v_t$. That is, it means $\nabla_{\theta_t} \mathcal{J}(\theta_{t+1})$, and this is an attempt to check the future slope in advance. Therefore, it is possible to reduce the phenomenon of not entering the minima and fluctuating the vicinity when it reaches the minima. RMSprop is summarized as Equations (15) and (16).

$$G_t = \gamma G_{t-1} + (1-\gamma)(\nabla_{\theta_t} J(\theta_t))^2 \quad (15)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} * \nabla_{\theta_t} J(\theta_t) \quad (16)$$

RMSprop is a method using an exponential moving average. In Adagrad, there was a problem that the learning rate decreased as the weight update increased. As a weight of (1-a) is added to the exponential moving average, the influence decreases over time. According to filter theory, these weights are called the forging factor or the decoding factor, and in equation (15), G_t is added as an exponential average. Adelta, like RMSprop, is a method designed to solve the problem of decreasing the learning rate in Adagrad, and the weights are updated as shown in Equations (17), 18, and based on Equations (19).

$$gt = \nabla_{\theta_t} J(\theta_t) \quad (17)$$

$$\Delta\theta_t = -\frac{\text{RMS}[\Delta\theta]_{t-1}}{\text{RMS}[g]_t} gt \quad (18)$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \quad (19)$$

Adelta uses $E[g^2]_t$ of Equation (20) to reduce the influence of the information by storing the expected value of the square of the gradient rather than storing the sum of the squares of the past gradient.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \quad (20)$$

Using this, the change in the current weight is defined as Equation (21)

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (21)$$

Since root mean square (RMS) is $\sqrt{\frac{x_1^2 + x_2^2 + \dots + x_n^2}{n}}$, it may be expressed as equation (22).

$$\Delta\theta_t = -\frac{\eta}{\text{RMS}[g]_t} g_t \quad (22)$$

Here, when first order method methods such as SGD, Momentum, and Adagrad are used, the unit of the weight and the changed weight becomes an reciprocal relationship as expressed in proportion in Equation (23). Therefore, the unit was matched by introducing the Hessian matrix, a second order method matrix, in Equation (22). It is expressed as Equation (24).

$$\text{units of } \Delta\theta \propto g \propto \frac{\partial J}{\partial \theta} \propto \frac{1}{\text{units of } \theta} \quad (23)$$

$$\Delta\theta \propto H^{-1} \propto \frac{\partial J / \partial \theta}{\partial^2 J / \partial \theta^2} \propto \text{unit of } \theta \quad (24)$$

Through the process of matching units, a Hessian matrix representing the characteristics of curvature was introduced, and a second order Taylor expansion that approximates the function to the second term is applied. Through this, we solved two problems, which were difficult to solve with first-order differentiation: the problem of determining whether the saddle point is the maximum or the minimum, and the learning rate is reduced.

III. Face recognition lightening model with mask wearing

1. Masked Face Recognition Algorithm Overview

In this study, we propose a lightweight algorithm by improving the MobileNetV3 used for existing face recognition. MobileNetV2 increased the accuracy by using the Squeeze and excitation module, but MobileNetV3 judged that there was an efficiency of the filter to judge the image in the last step. As the number of intermediate layers increases, the number of parameters of the model also increases, so the SE block is removed and the nonlinear function H-Swish and the linear function ReLU6 are applied. The idea of the proposed algorithm aims to maintain accuracy while reducing the number of layers

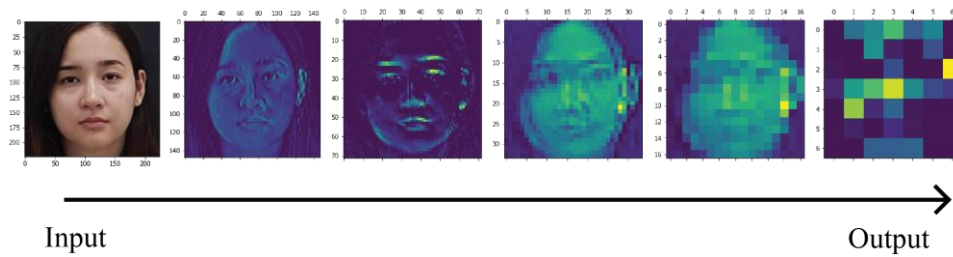


Figure 9. Image with filter layer applied to face image

Visualizing the layers of V3 is shown in Figure 9. As follows, you can see the shape of a person at the beginning of the layer, but it becomes more difficult to recognize the shape toward the last layer. And when V3-Small was applied to the Same network as it was, different results were shown. The following experiment was conducted to apply V3-Small to the Same network as it was and compared with the accuracy of the proposed network with the intermediate layer further removed.

Table 2. Comparison of parameters, amount of computation, and accuracy according to H-Swish function and ReLU6

	Parameters	Accuracy	MAdds
HS -1 // ReLu6 – 1 (Proposed Network)	4,520	96.56%	4,982
HS -2 // ReLu6 – 2	79,781	94.97%	643,307
HS -3// ReLu6 – 3	115,725	94.18%	933,201
HS -5// ReLu6 – 3	142,621	91.01%	1,149,993
HS -6// ReLu6 – 3	302,435	88.10%	2,388,364
HS -8// ReLu6 – 3 (MobileNetV3-Small)	1,594,543	56.35%	2,406,151

Table 2 is a comparison of the accuracy when reducing the hard-swish HS and ReLU6 in Bottleneck. The accuracy of the general V3 was 56.35%, and the accuracy of the proposed network became 96.56%. Unlike EfficientNet [36], etc., in the Siamese network, there is a phenomenon in which the accuracy decreases as the depth of the network increases. This phenomenon is due to the characteristics of the Siamese network. This is because the Siamese network created to learn a small number of images occurs in the process of generalizing

image features as the depth of the network increases. Therefore, the network proposed in this study used 1 HS and 1 ReLU6 function.

2. Layer of the mask-wearing model

100 x 100 pixel image pairs composed of 3 channels are entered through the Input Layer. After performing a 2D convolution operation, the pixels enter a convolution block that performs batch normalization. The number of filters used here is 16, and the number of Kernel is set to 3 x 3. While performing the convolution operation, the number of pixels to skip is calculated by jumping two horizontally and vertically. The H-Swish function was used as the activation function used in the first convolution block. After this, the pixels enter the Bottleneck block. This block includes the process of performing a global average pooling process on the network block shown in Figure 3. First, a convolution operation is performed with the same filter and kernel, and a Depthwise convolution operation that synthesizes only one channel with one filter is performed. After normalization of pixels for each batch is performed, the ReLU6 activation function is performed. Next, it goes through a compression process called Squeeze, which is a process of global average pooling by extracting the average of pixel values on the feature map to make an image tensor a vector, then going through the ReLU, Hard sigmoid function to make it a 1 x 1 scalar, and then multiplying it with the image that came into the input. Here, the ReLU and Hard sigmoid functions are expressed in the following equations (25) and (26).

$$ReLU = \max(0, x) \quad (25)$$

$$Hard\ sigmoid = \max(0, \min(1, \frac{(x+1)}{2})) \quad (26)$$

In the second Bottleneck block, if the Global average pooling process is performed after using 24 filters and hard-swish functions, the proposed network is completed as shown in Table 3.

Table 3. Detailed configuration diagram of the proposed network

Layer	Out Shape	Param #	Connected to
inputLayer	(None,100,100,3)	0	
conv2d	(None,50,50,16)	448	inputLayer
batch normalization	(None,50,50,16)	64	conv2d
activation	(None,50,50,16)	0	batch normalization
conv2d	(None,50,50,16)	272	activation
batch normalization	(None,50,50,16)	64	conv2d
activation	(None,50,50,16)	0	batch normalization
depthwise_conv	(None,50,50,16)	160	activation
batch normalization	(None,50,50,16)	64	depthwise_conv
activation	(None,50,50,16)	0	batch normalization
global average pooling	(None,16)	0	activation

dense	(None,16)	272	global average pooling
dense	(None,16)	272	dense
reshape	(None,1,1,16)	0	dense
mutiply	(None,50,50,1 6)	0	reshape
conv2d	(None,50,50,1 6)	272	mutiply
batch normalization	(None,50,50,1 6)	64	conv2d
add	(None,50,50,1 6)	0	batch normalization
conv2d	(None,50,50,2 4)	408	add
batch normalization	(None,50,50,2 4)	96	conv2d
activation	(None,50,50,2 4)	0	batch normalization
depthwise_conv	(None,50,50,2 4)	240	activation
batch normalization	(None,50,50,2 4)	96	depthwise_conv
activation	(None,50,50,2 4)	0	batch normalization
global average pooling	(None,24)	0	activation
dense	(None,24)	600	global average pooling
dense	(None,24)	600	dense
reshape	(None,1,1,63)	0	dense
mutiply	(None,50,50,6)	0	activation

	3)		dense
conv2d	(None,50,50,6 3)	600	mutiply
batch normalization	(None,50,50,6 3)	96	conv2d
conv2d	(None,50,50,6 3)	800	batch normalization
batch normalization	(None,50,50,6 3)	96	conv2d
activation	(None,50,50,6 3)	0	batch normalization
global average pooling	(None,63)	0	activation
Total params	4,840		
Trainable params	4,520		
Non-trainable params	320		

3. Face authentication process for mask-wearing models

The model applied in this study bundles two networks to apply weight sharing. After that, the overall process of classifying facial images is shown in Figure 10.

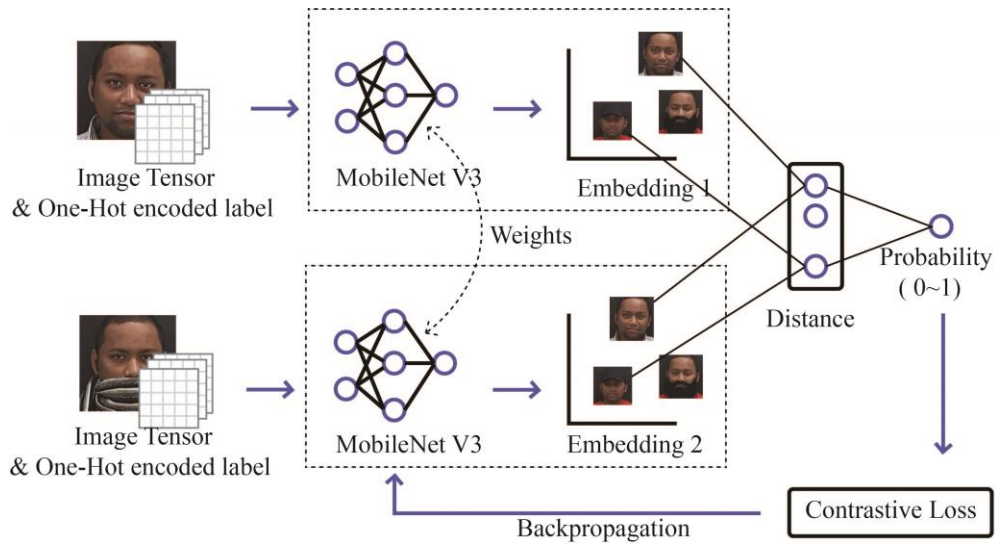


Figure 10. Proposed Network's Facial Image Classification Process

First, put the pixel values of the training images in the array. The labels are also put in the array in the order of the images. Next, put the pixel values and labels of the test images in the array. Normalize to a value between 0 and 1 to prevent the influence of a specific feature with a large scale from increasing and to improve the learning speed. Next, create pairs of images to put images into the two Siemens networks. One-shot learning is applied, so one pair consists of two images. If the two images are the same, they become positive pairs, and if the two images are different, they become negative pairs.

The face image of the same person is included in the positive pair, and the face image of another person is included in the negative pair. In order to put image pixels in the network, the negative pair is included in the odd number array, and the positive pair is included in the even number array. The images are entered one by one in each of the two proposed networks. Like grouping the network of the GAN, the end of the two networks is grouped together, and then as the next layer, a distance layer that can measure the distance between images entering the two networks is added. The model learns by repeating the process of updating the weights through backpropagation of the images. In this study, the larger the Euclidean distance calculated in the distance layer, the larger the value of 'Dissimilarity' came out, so that the performance of the model could be visually confirmed.

IV. Evaluation of the proposed model

1. Face image database

The Sejong Face Database: A Multi-Modal Disguise Face Database [37] used in this study consists of 44 men and 26 women. Their ages range from 22 to 50. Race is 30 percent Middle Eastern, 28 percent Southeast Asian, 20 percent Caucasus, 16 percent Russian, and 6 percent African. Each person consists of images of people disguised as glasses, hats, scarves, wigs, and more, as well as images of a normal face without anything on. Details are shown in Table 4.

Table 4. Disguised Types of Sejong Face Database

Type	Gender	
	Male	Female
Natural Face	<input type="checkbox"/>	<input type="checkbox"/>
Real beard	<input type="checkbox"/>	
Cap	<input type="checkbox"/>	<input type="checkbox"/>
Scarf	<input type="checkbox"/>	<input type="checkbox"/>
Glasses	<input type="checkbox"/>	<input type="checkbox"/>
Mask	<input type="checkbox"/>	<input type="checkbox"/>
Make up		<input type="checkbox"/>
Wig		<input type="checkbox"/>
Fake beard	<input type="checkbox"/>	
Fake mustache	<input type="checkbox"/>	
Wig-Glasses		<input type="checkbox"/>

Wig-Scarf		<input type="checkbox"/>
Cap-Scarf	<input type="checkbox"/>	<input type="checkbox"/>
Glasses-Scarf	<input type="checkbox"/>	<input type="checkbox"/>
Glasses-Mask	<input type="checkbox"/>	<input type="checkbox"/>
Fake Beard-Cap	<input type="checkbox"/>	
Fake Beard-Glasses	<input type="checkbox"/>	

In Table 4, Wig-Glasses means that the wig and glasses are worn together. It means that the Fake beard-Cap is also wearing fake wigs and glasses together. Men do not include images of makeup, and women do not include images disguised as fake beards. The Natural Face means a general face image without camouflage. The general face image gazes in a total of four directions: the face looking at the front, the face looking at the left, the face looking at the right, and the face looking at the bottom. One face image has a total of four types of images that combine infrared images, thermal images, visible images, and visible images with infrared rays. The camera resolution of the infrared image is 1,680 1050, and it has a frequency of 770 nm to 1,000 nm. The camera resolution of the thermal image is 768756, and it has a frequency of 750 nm to 1,400 nm. The camera resolution of the visible image is 4,032 3,024, and it has a frequency of 400 nm to 770 nm.

Here, 63 faces were extracted from visible light images. Gender consisted of 21 women and 42 men. The training dataset consists of a total of 7,560 images, using images with a face exposed and images with a face covered with hats, scarves, and glasses. The image with a face exposed within one face class exists between about 15 at the same angle as the original database. About 50 images with a face covered with hats, scarves, and glasses are included. The test data used images with a face covered by wearing a mask. It consists of a total of 315 images, with about 5 images per person. The image is shown in Figure 11



Figure 11. Train Data (top) and Test Data (bottom) images used for learning

To make these images into the Siamese Network, they are made in the form of a positive pair and a negative pair. The positive pair is the grouping of training data and training data into the same face image, and the negative pair is the grouping of training data and training data into different face images. Figure 12 is a visualization of the pairs.

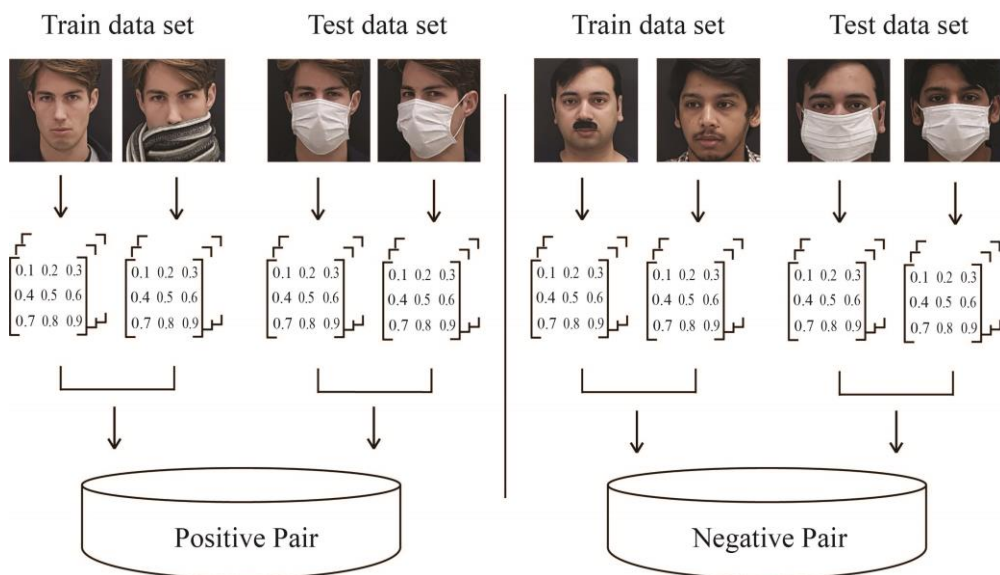


Figure 12. Formation of Positive Pair and Negative Pair

2. How to evaluate the performance of the model

A method of evaluating the performance of an algorithm using Precision and Recall is applied to various fields. Using this, TP (True Positive), FP (False Positive), TN (True Negative), and FN (False Negative) may be obtained. Terms and definitions are specified in Tables 5 and 6

Table 5. Confusion matrix for classification algorithm evaluation

	Predicted Class		
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Table 6. Confusion matrix components for classification algorithm evaluation

Term	Definition
TP	An accurate prediction of the answer
FP	Predicting the wrong answer as the correct answer (Error)
FN	An accurate prediction of the wrong answer
TN	Predicting the correct answer to be incorrect (Error)

When evaluating a machine learning model, the method of evaluating with the confusion matrix as above is a reasonable evaluation method. In order to evaluate the model based on the Siemens network, AUC and Recall (=Sensitivity) were mainly evaluated. Recall is the ratio of what the model predicts as the correct answer among the actual correct answers because if the two pictures are the same, the distance shows a close value, and if the other picture shows a high value. The equation for calculating Recall is as in Equation (27).

$$Recall = \frac{TP}{TP + FN} \quad (27)$$

Specificity refers to the rate at which the actual wrong answer is judged to be the wrong answer. False Positive Rate (FPR) is expressed as Equation (28) as the ratio between the actual incorrect answer and the model's incorrect prediction as the correct answer, and is the same as 1-specificity. In general, the ROC curve is a graph showing FPR and the sensitivity, True Positive Rate (TPR), on the x and y axes.

$$\begin{aligned} FPR &= \frac{FP}{FP + TN} = \frac{(TN + FP) - TN}{TN + FP} \\ &= 1 - \frac{TN}{TN + FP} = 1 - \text{Specificity} \end{aligned} \quad (28)$$

AUC is a value representing the area below the ROC curve. It can be seen that the closer to 1, the higher the performance of the model. To obtain AUC, an appropriate threshold should be selected from the ROC curve. The threshold is too low and the precision is high, but the false negative Type 2 Error occurs, and if the threshold is too high, the true negative Type 1 Error occurs. In this study, the target false positive rate was set to 0.003 so that an appropriate threshold was selected.

3. Result

The performance of the model is evaluated in three aspects. First, we evaluated how well this model could distinguish other classes, such as regular face and scarf hat, in addition to masks. Second, we evaluated how efficiently the proposed model was calculated and the model with high accuracy by comparing it with other models. Third, it was confirmed to what extent the performance deviation changed when other optimization functions were applied.

First, the model tested not only the face with a mask, but also the face without anything, the face with a scarf, and the face with a hat. Each of the four classes was trained 100,000 times. The results are from Table 7 and Table 8.

Table 7. AUC of models with different classes other than mask-wearing face images

Iteration	AUC			
	Classes			
	Normal	Mask	Scarf	Cap
10,000	0.83	0.96	0.95	0.89
20,000	0.86	0.90	0.97	0.93
40,000	0.93	0.96	0.98	0.95
60,000	0.93	0.96	0.98	0.96
80,000	0.93	0.97	0.98	0.97
100,000	0.93	0.97	0.99	0.97

Table 8. Sensitivity of models with classes other than mask-wearing face images

Iteration	Sensitivity			
	Classes			
	Normal	Mask	Scarf	Cap
10,000	11.4 %	29.8%	19.4%	17.0%
20,000	13.3 %	22.4%	22.8%	30.2%
40,000	20.9 %	26.5%	30.0%	28.7%
60,000	23.2 %	32.2%	33.9%	32.4%
80,000	23.3%	31.0%	39.5%	38.6%
100,000	27.1%	37.8%	40.9%	40.2%

The class that showed the lowest AUC value was 0.93, for normal faces. The model judged the face AUC value of the person wearing the scarf most accurately as 0.99. Next, the face of the person wearing the mask and the face of the person wearing the scarf are judged to be 0.97. The sensitivity of the face of the person wearing the mask was 37.8%, and the face of the person wearing the hat was 40.2%, which judged the face of the person wearing the hat a

little better. In all four cases, it was confirmed that the model worked well when we saw that the AUC value was close to 1. Figure 13, Figure 14, Figure 15, and Figure 16 show the ROC curve at the final iteration.

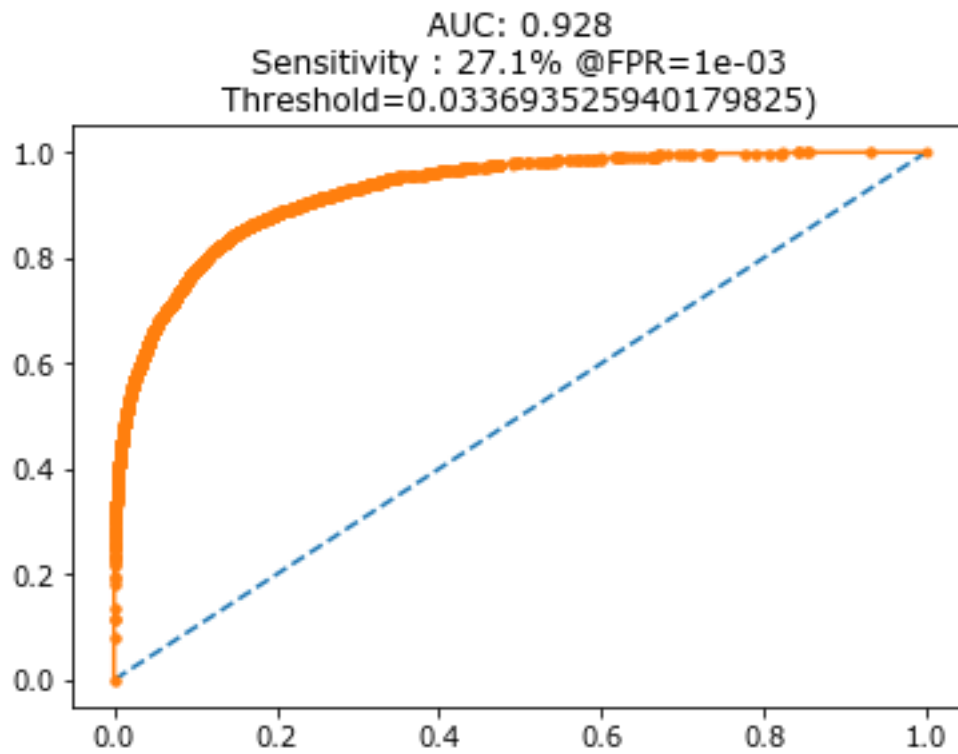


Figure 13 In the final iteration. ROC curve when applying an image with unblocked face with test data

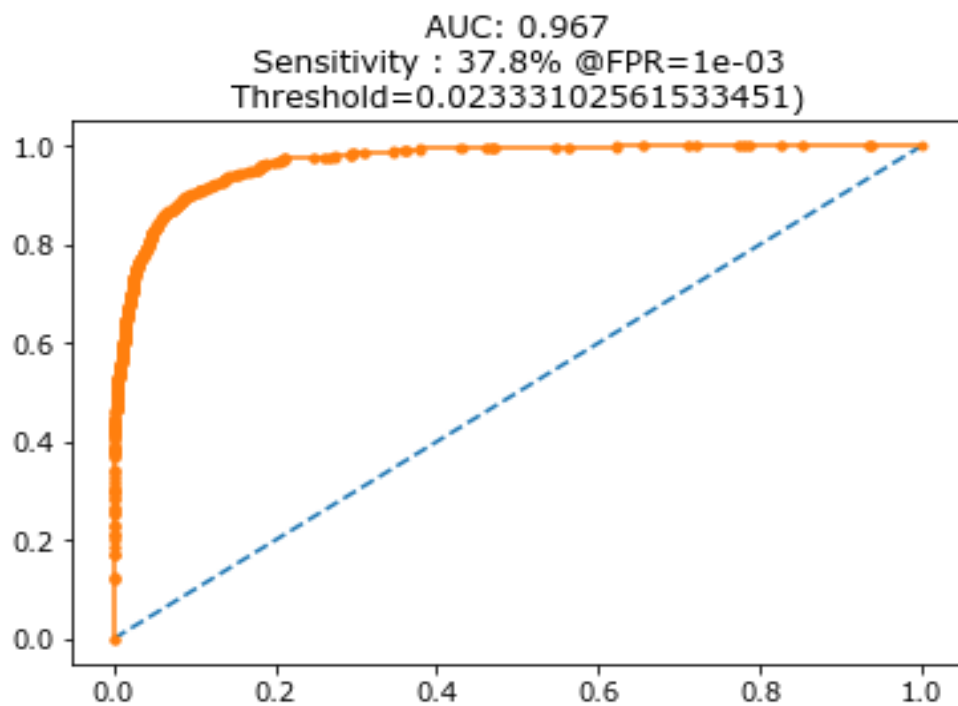


Figure 14. ROC curve when the image wearing a mask is applied as test data in the final iteration

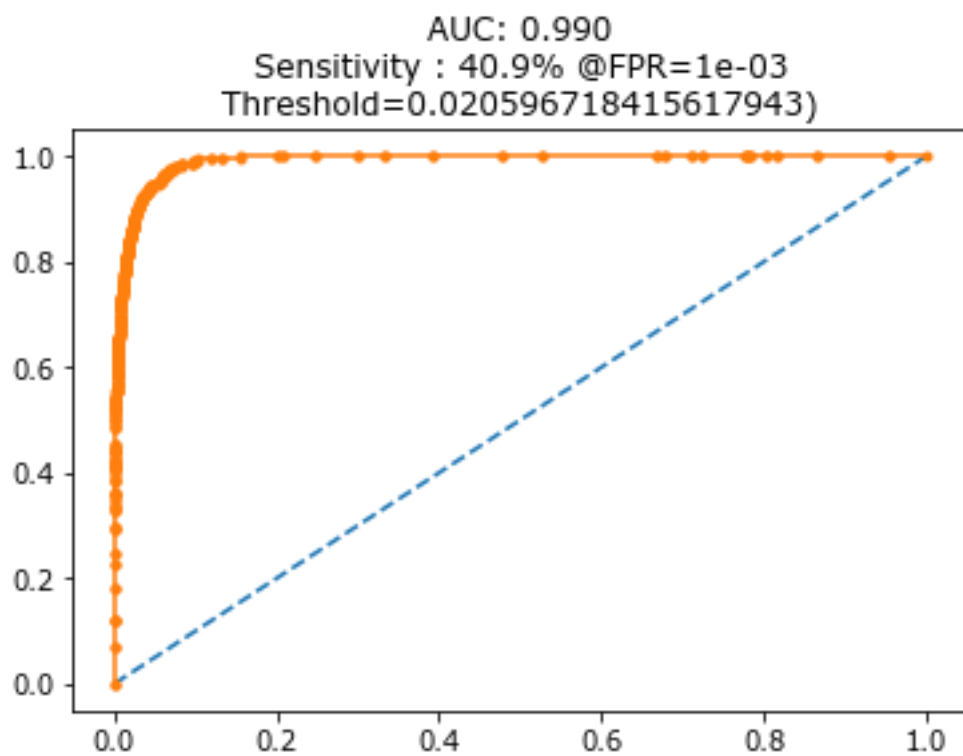


Figure 15. ROC curve when applying the image of wearing a scarf as test data in the final iteration

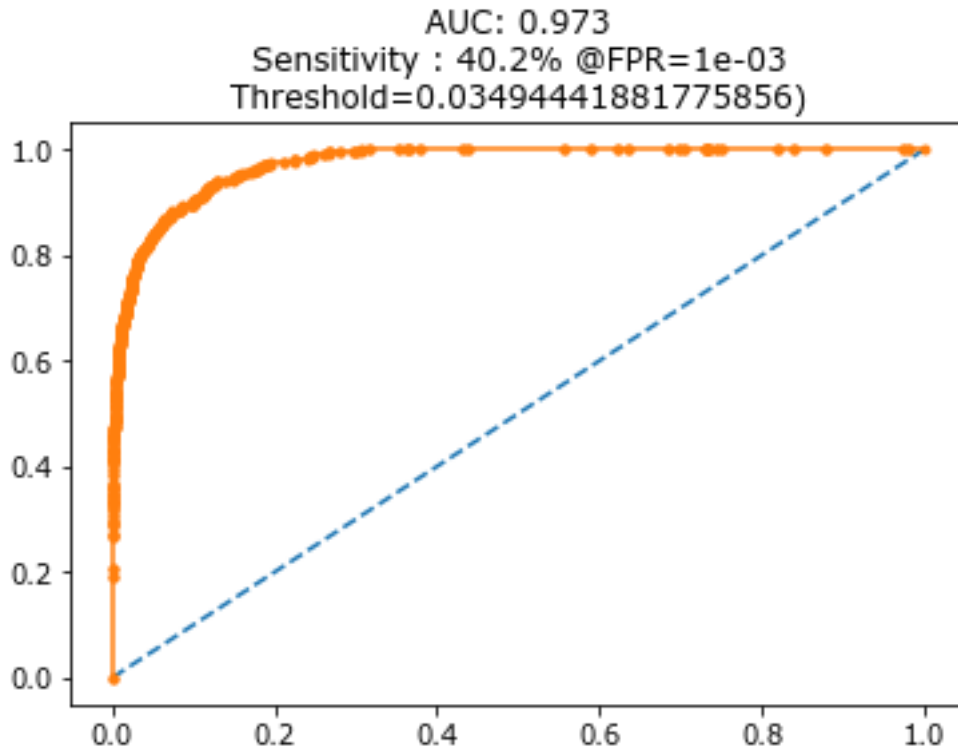


Figure 16. ROC curve when the image of wearing a hat is applied as test data in the final iteration

Next, we confirmed how efficient the performance of this model is compared to other models. A model was used to add layers Arcface[39] and Sphreface[40] which are functions based on distance learning to Resnet[38]. In order to confirm the basic performance of the Siemens Net, a model consisting of only Global average pooling and activation function layers using ReLU was used. MobileNet V1, V2, V3-small was put into the Siemens network and tested to compare how different the amount of computation is from the proposed network. Models are compared in terms of the number of parameters and multiplex adds (MAdS), speed, and acuity. Table 9 shows the results of the experiment.

Table 9. Performance Comparison between Proposed Network and Other Models

Network	Parameters	MAdds	Speed	Accuracy
Resnet18 - Arcface	24,418,621	505,465,454	360 ms	98.44%
Resnet18 -Sphereface	24,418,621	505,465,454	360 ms	98.96%
Siamese-Net -AlexNet	25,151	17,892	18 ms	61.38%
Siamese -MobileNetV1	3,293,439	26,238,119	48 ms	82.28%
Siamese -MobileNetV2	2,304,575	18,539,071	78 ms	76.19%
Siamese -MobileNetV3 -small	1,594,543	12,691,962	76 ms	56.35%
Siamese -MobileNetV3 (Proposed method)	4,840	4,982	11 ms	96.56%

Among the experimental results, the model that applied V3-small to the Simese Network recorded 56.35% of accuracy. The model that applied AlexNet to the Simese Network showed 61.38%. Arcface using ResNet showed 98.44% accuracy and 98.96% accuracy when Sphereface was applied. Proposed Network showed 96.56% accuracy. It was confirmed that the accuracy was slightly lower, but the performance was good in terms of the amount of computation. As shown in Figure 17, when using Arcface and Sphereface, it was confirmed that there are about 24 million parapeters and about 500 million MAdds, the amount of computation. The learning speed was 360 ms per epoch. In the case of the proposed network,

it has about 5,000 parameters, MAdds about 5,000 came out, and the learning was completed in 11 ms. Therefore, it was confirmed that efficient classification can be performed with a very small capacity if it is not connected to the embedded system, the Internet, or if it is built in using an application on the mobile.

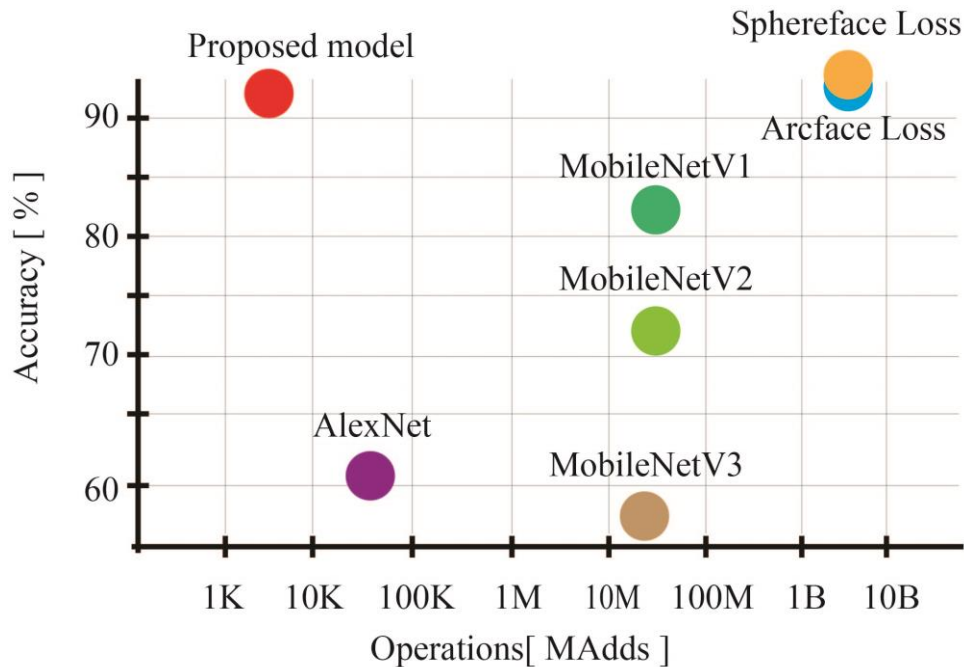


Figure 17. Comparison of the amount of computation and accuracy of the model

Next, the difference in performance was confirmed when another optimization function was applied to the proposed model. The experiment was conducted with 100, 300, 500, 700, and 1000 epochs respectively to compare the optimization functions. By checking the learning speed through this, the most appropriate optimization function is identified in order to learn the model efficiently, and the function that reliably converges to minima is identified. In addition, by classifying and comparing the Momentum-based optimization algorithm and the step size-focused line search optimization algorithm, the effect of the method of storing the direction vector and heading to minima on the proposed model was confirmed.

Table 10. Momentum based optimization accuracy comparison according to Epoch

	100 epoch	300 epoch	500 epoch	700 epoch	1000 epoch
Momentum	90.28%	88.69%	88.13%	88.80%	89.37%
NAG	88.16%	89.07%	89.43%	89.66%	89.82%

In the case of Momentum, the highest accuracy of 90.28% was achieved at 100 epochs according to the concept of finding the direction vector and quickly heading to minima. In the case of learning up to 300 epochs and 500 epochs, it was confirmed that inertia occurred due to the calculation of the wrong direction vector, and the accuracy decreased as the learning increased. In the case of NAG, it was confirmed that the highest accuracy was achieved at 1000 epochs by preventing the occurrence of inertia in the wrong direction by applying the method of calculating the gradient after moving one step. Table 11 is the accuracy of the optimization functions to find minima involved in adjusting the size of the step.

Table 11. Comparison of Line Search Optimization Accuracy according to Epoch

	100 epoch	300 epoch	500 epoch	700 epoch	1000 epoch
SGD	78.00%	78.44%	81.10%	81.19%	81.27%
RMSProp	98.89%	99.57%	99.58%	99.71%	99.84%
Adadelta	76.97%	78.75%	80.61%	79.17%	78.45%

In the case of SGD optimizer, one training data is used to go one step in one of the basic Gradient Descent methods. Other epochs show similar patterns, but when looking at the results based on 1000 epochs, it can be seen that the learning proceeds slowly among algorithms other than the Adatelta algorithm and shows the lowest accuracy. In the case of RMSprop, it shows the highest accuracy, and it can be seen that it is stably oriented from 100 epochs to 1000 epochs. In the case of Adatelta, optimization is performed using the Hessian matrix together with the Jacobian matrix. However, it was confirmed that the accuracy was the highest when learning up to 500 epochs, but the accuracy decreased when further learning was performed. One of the problems of the second order method was the problem of not converging stably. Finally, using the Proposed Network, the results of expressing the degree to which the model recognizes the difference in face between the two images in Figure 18 and Figure 19 were visually confirmed.




	
Person Num.	07-56 26-00 38-38 28-06 17-06
Dissimilarity	1.334 2.003 0.009 1.285 1.184
	
Person Num.	14-14 01-01 43-02 21-08 25-25
Dissimilarity	0.200 0.204 1.707 1.269 0.061
	
Person Num.	18-07 53-00 16-16 48-48 59-59
Dissimilarity	1.536 1.785 0.015 0.038 0.035

Figure 18. The result of quantifying the difference in similarity according to the facial image1




	
Person Num.	56-02 33-33 60-60 52-09 08-08
Dissimilarity	1.638 0.158 0.013 1.546 0.217
	
Person Num.	31-31 04-03 04-04 50-07 35-00
Dissimilarity	0.016 0.903 0.168 1.447 1.925
	
Person Num.	40-04 27-27 24-24 06-03 18-18
Dissimilarity	1.430 0.018 0.022 0.493 0.016

Figure 19. The result of quantifying the difference in similarity according to the facial image
2

V. Conclusions and Future Research Direction

Face recognition was showing good performance while wearing a mask. However, due to the pandemic, wearing a mask has become more common. Therefore, a system that can recognize the face even if only a specific part of the face is visible while wearing a mask is needed. In this study, we proposed a mask-wearing face recognition deep learning model that can effectively perform even on embedded devices with limited computational power such as mobile devices.

In this study, a model to which the concept of sharing weights was applied was introduced based on MobileNet focusing on light weight. By combining the core concepts of the two networks, the amount of computation could be reduced while achieving 96.56 percent accuracy. With this model, the observation was conducted by dividing it into two functions focusing on the step size and functions focusing on the direction of the vector. Functions applying the Jacobian matrix showed better performance than Adelta applying the Hessian matrix. Among them, it was confirmed that RMSprop achieves the fastest and highest accuracy among functions. Furthermore, the test data was tested by changing the image of the face with a scarf, the face with a hat, and the face image without anything, not the face with a mask. In other images, it was confirmed through AUC that the model showed robust performance. Compared to the model to which the SOTA functions of the distance-based learning method were applied, the accuracy was slightly lower than that of the model to which the SOTA functions of the distance-based learning method were applied, but considering the amount of computation of the model, the proposed model is an efficient model because it is a very lightweight model.

Therefore, a model using a small number of parameters, such as the model of this study, can be applied to embedded devices with limited computational power. In other words, when installed in a mobile device such as a mobile phone, the capacity of the installation program can be drastically reduced. In addition, it was verified that the proposed model can be applied not only to the face of the person wearing the mask, but also to the face covered with a scarf or glasses. The performance of the model according to the learning time was confirmed by applying various optimization functions. Even when the performance of the proposed model is compared to the performance of other models, it showed high accuracy, but there is a problem that it is not the highest accuracy. If a distance-based learning method that is focused on a little more accuracy is applied, the amount of computation will increase, but the problem can be overcome. In this study, only visual images were used, but if images of thermal imaging cameras and infrared cameras are applied in the future based on this, there would be no problem in applying them to the face disguise recognition system applied to the immigration system. In addition, if the SIFT algorithm and Advost algorithms used in the existing disguise recognition system are added to the model, a more robust system can be made.

References

- [1] Anil K. Jain, Arun Ross, Salil Prabhkar, "An Introduction to Biometric Recognition", IEEE Transactions on Circuits And Systems For Video Technology, Vol. 14, no. 1, pp. 4-20, Jan 2004
- [2] Asadullah Laghari, Waheed-ur-Rehman, Dr. Zulfiqar Ali Memon, "Biometric Authentication technique Using Smartphone Sensor", Proceedings of 2016 13th International Bhurban Conference on Applied Sciences & Technology(IBCASP), pp. 381-384.
- [3] Wang, Mei, and Weihong Deng. "Deep face recognition: A survey." arXiv preprint arXiv:1804.06655 (2018).
- [4] Borghi, Guido, et al. "Poseidon: Face-from-depth for driver pose estimation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
- [5] Oh Song, Hyun, et al. "Deep metric learning via lifted structured feature embedding." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [6] Zhang, Zheng, et al. "A survey of sparse representation: algorithms and applications." IEEE access 3 (2015): 490-530.
- [7] Wen, Chenglu, Daniel E. Guyer, and Wei Li. "Local feature-based identification and classification for orchard insects." Biosystems engineering 104.3 (2009): 299-307.
- [8] Fogel, Itzhak, and Dov Sagi. "Gabor filters as texture discriminator." Biological cybernetics 61.2 (1989): 103-113.

- [9] Zhang, Guangcheng, et al. "Boosting local binary pattern (LBP)-based face recognition." Chinese Conference on Biometric Recognition. Springer, Berlin, Heidelberg, 2004.
- [10] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.
- [11] Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
- [12] Zhou, Shengwei, et al. "Double Additive Margin Softmax Loss for Face Recognition." Applied Sciences 10.1 (2020): 60.
- [13] Wang, Dayong, Charles Otto, and Anil K. Jain. "Face search at scale." IEEE transactions on pattern analysis and machine intelligence 39.6 (2016): 1122-1136.
- [14] Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). Vol. 2. IEEE, 2006.
- [15] Deng, Jiankang, et al. "Arcface: Additive angular margin loss for deep face recognition." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
- [16] Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). Vol. 2. IEEE, 2006.

- [17] Bromley, Jane, et al. "Signature verification using a "siamese" time delay neural network." *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993): 669-688.
- [18] Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." *ICML deep learning workshop*. Vol. 2. 2015.
- [19] Jeong, Yoosoo, et al. "Accurate age estimation using multi-task Siamese network-based deep metric learning for frontal face images." *Symmetry* 10.9 (2018): 385.
- [20] Zhang, Cheng, et al. "Siamese neural network based gait recognition for human identification." *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.
- [21] Shen, Jianbing, et al. "Visual object tracking by hierarchical attention siamese network." *IEEE transactions on cybernetics* 50.7 (2019): 3068-3080.
- [22] Mueller, Jonas, and Aditya Thyagarajan. "Siamese recurrent architectures for learning sentence similarity." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. No. 1. 2016.
- [23] Wang, Yaqing, and Quanming Yao. "Few-shot learning: A survey." (2019).
- [24] Li, Hongyang, et al. "Finding task-relevant features for few-shot learning by category traversal." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019.
- [25] Large Margin Deep Networks for Classification
- [26] Kaya, Mahmut, and Hasan Şakir Bilge. "Deep metric learning: A survey." *Symmetry* 11.9 (2019): 1066.

- [27] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
- [28] Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
- [29] Tan, Mingxing, et al. "Mnasnet: Platform-aware neural architecture search for mobile." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
- [30] Howard, Andrew, et al. "Searching for mobilenetv3." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.
- [31] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011).
- [32] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).
- [33] Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).
- [34] Qian, Ning. "On the momentum term in gradient descent learning algorithms." Neural networks 12.1 (1999): 145-151.
- [35] Nesterov, Yurii. "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$." Doklady an ussr. Vol. 269. 1983.

[36] Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International Conference on Machine Learning. PMLR, 2019.

[37] Cheema, Usman, and Seungbin Moon. "Sejong face database: A multi-modal disguise face database." Computer Vision and Image Understanding 208 (2021): 103218.

[38] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[39] Deng, Jiankang, et al. "Arcface: Additive angular margin loss for deep face recognition." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.

[40] Liu, Weiyang, et al. "Sphereface: Deep hypersphere embedding for face recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.