Universal Face Recognition Using Multiple Deep Learning Agent and Lazy Learning Algorithm

Kenny Vincent[1] and Yosi Kristian[2*]
[1−2]Informatics Department, Institut Sains dan Teknologi Terpadu Surabaya
Jawa Timur 60284, Indonesia
Email: [1]vincent.kenny98@gmail.com, [2]yosi@stts.edu

***Abstract*—Mainstream face recognition systems have a problem regarding the disparity of recognizing faces from different races and ethnic backgrounds. This problem is caused by the imbalances in the proportion of racial representations found in mainstream datasets. Hence, the research proposes using a multi-agent system to overcome this problem. The system employs several face recognition agents according to the number of races that are necessary to make data encodings for the classification process. The first step in implementing this system is to develop a race classifier. The number of races is arbitrary or determined differently in a case-by-case scenario. The race classifier determines which face recognition agent will try to recognize the face in the query. Each face recognition agent is trained using a different dataset according to their assigned race, so they have different parts in the system. The research utilizes lazy learning algorithms as the final classifier to accommodate a system with the constant data flow of the database. The experiment divides the data into three racial groups, which are black, Asian, and white. The experiment concludes that dividing face recognition tasks based on racial groups into several face recognition models has better performance than a single model with the same dataset with the same imbalances in racial representation. The multiple agent system achieves 85% on the Face Recognition Rate (FRR), while the single pipeline model achieves only 80.83% using the same dataset.**

***Index Terms*—Face Recognition, Multiple Deep Learning Agent, Lazy Learning Algorithm**

I. Introduction

FACE recognition systems encounter similar problems regarding recognizing faces from lighter-skinned subjects better than their darker-skinned counterparts. It is caused by the racial representation ratio of mainstream datasets used by big companies. Mainstream face image datasets contain a majority of white-skinned individuals. This imbalance in racial representation causes what is called Own Race Bias (ORB). ORB refers to the tendency of being more accurate at recognizing faces of one race than faces of another race [1]. In this case, the face recognition systems of "own race" are the most represented group of the racial background of the datasets.

This problem is rendered unfixable without balancing the racial representation of the used datasets if people want to preserve the mainstream way of how face recognition systems work. A recent analysis discovers that the composition of mainstream datasets is highly imbalanced [2]. Along with those imbalances comes a huge gap of recognition ability in different race and gender groups. One of the largest gaps is between recognizing white women with 94% accuracy and black women with only 65.5% accuracy on a dataset.

Previous research uses Convolutional Neural Network (CNN) to directly recognize the identity of the given queries [3]. Another research applies a lazy learning algorithm which is a k-Nearest Neighbor (kNN) in particular, to perform a better face recognition task [4]. A lazy learning algorithm receives a face encoding as an input to be estimated to produce a prediction of the identity.

Instead of using a standalone model like a basic CNN recognition system, the research proposes using a multiple-agent CNN feature extractor for each race group. The number of the racial group is determined arbitrarily to suit the need of the system. For each race group, a different feature extractor will be employed. Therefore, each feature extractor can focus on acquiring a face encoding of "its own" race environment. A lazy learning algorithm is employed to accommodate the flexibility the system necessitates to withstand a constant flow of datasets.

In the research, face images are used to determine

Received: Sept. 04, 2020; received in revised form: Feb. 08, 2021; accepted: Feb. 10, 2021; available online: Aug. 30, 2021.
*Corresponding Author

broadly generalized race groups and face encoding of the individual. It uses deep learning to develop a race classifier that categorizes individuals into three groups: black, Asian, or white. These groups are arbitrarily selected to demonstrate how a multi-pipeline model will perform differently from a single-pipeline model with an imbalance of the racial representation in the dataset. It experiments with Convolutional Auto-Encoder (CAE) and various pre-trained face recognition models to extract features of the face. Then, transfer learning is performed to the pre-trained models to suit the need of the system. It uses lazy learning classifiers like kNN and Gaussian Naïve-Bayes (GNB) to be the last classifier to produce the predicted identity of the query.

There are several key contributions to the research. First, the research integrates a race classifier into a face recognition system to boost the performance of the system. Second, it creates a multi-way face recognition system rather than the standard single-way mainstream systems to combat the problem of racial representation imbalance in datasets. Third, the researchers analyze the use of CAE for face recognition purposes. Last, a flexible network is developed to overcome various data flow of the dataset while using the multiple-agent method to optimize the accuracy of the system.

II. Related Work

A. Convolutional Neural Network (CNN), Face Detectors, and Race Classifiers

Deep CNN has been at the forefront of computer vision for its promising potential. Reference [5] proposes a novel feature in the learning method for halftone image classification with great performance. The method applies stacked Sparse Auto-Encoders (SAE) to encode or extract features of halftone images by using softmax regression to classify halftone images. Another previous research proposes a CNN model for multilabel image classification called Hypotheses-CNN-Pooling [6].

In the application of face-related purposes, face detection has been a major breakthrough. Reference [7] implements Histograms of Oriented Gradients (HOG) to perform face detection and face recognition. Face detection is performed using cascading windows to search for several face landmarks. HOG method is improved even further using methods like an ensemble of regression trees to estimate face landmark positions directly from a sparse subset of pixel intensities to achieve high-quality predictions. Moreover, face alignment boosts the performance of doing a face-related task. Meanwhile, another previous research has a deep neural network model with facial landmark features and recurrent regression to perform face alignment [8]. Using face alignment makes a search for faces in the image far easier because all the important landmarks on the face are rotated to match the standard triangle template.

Next, an application of race recognition is proposed [9]. The proposed system applies two architectures to the experiment to recognize if the face in the query is Vietnamese or from other races. Similarly, other researchers propose a novel approach to solve the problem of ethnicity classification using three classifiers with the same input to classify Chinese, black, and white people [10]. The research proves that CNN can perform well to classify race from image inputs.

B. Convolutional Auto-Encoder (CAE)

Auto-encoder is an architecture consisting of an encoder that compresses the input and a decoder to deconstruct the compressed value to reconstruct the input. Reference [11] introduces this method and lays out all the groundworks. It is inspired by another research [12] to implement an auto-encoder to extract features from human face images using supervised methods. An analysis of the usage and architecture of CAE was released in 2019 [13]. The analysis concludes that auto-encoders and their convolutional variants play a vital role in the current deep learning toolbox. The analysis also observes that CAE does not learn to copy the input. Thus, the number of channels in the bottleneck is not as important as the height and width of the bottleneck. Reference [14] suggests a novel approach using model-based CAE to overcome challenges in reconstructing 3D face images.

C. Transfer Learning

Transfer learning refers to a process of using an already-trained network to perform another task other than what it first intends to. However, it is still related enough to use the knowledge that it already has. The modern transfer learning method uses the inductive method [15] that the source and target domains are the same, but the source and target tasks are different from each other. In the inductive method, the algorithms try to utilize the inductive biases of the source domain to improve the target task.

Because of the widely available pre-trained CNN models, modern deep CNN rarely trains the entire network from the dataset only. Some of the available networks are VGG [16] and Xception [17]. In previous research, many pre-trained CNN models are adopted into a fully convolutional network for segmentation tasks [18]. Among many available models, research concludes that performing transfer learning for classifying purposes with VGG is more applicable than other models [19].

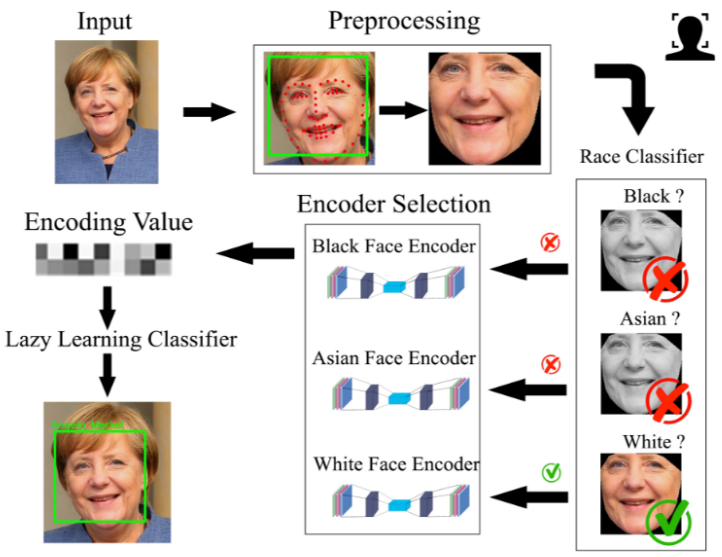

Fig. 1. The proposed architecture to reduce racial bias in face recognition.

D. Lazy Learning

Lazy learning is a type of algorithm that does the needed estimation when input is given. Contrary to eager learning like an artificial neural network, lazy learning algorithms do not perform any kind of training before receiving input. The most famous example of a lazy learning algorithm is the kNN algorithm. The algorithm was proposed by Thomas Cover and Peter Hart in 1967 [20]. The algorithm works by measuring a distance between the input and all of the data on the dataset and classifies it as the majority population of a certain number of neighbors that are the closest to it.

Reference [4] uses kNN to perform face recognition tasks. A more complex approach has been proposed to maximize the utilization of color information in input images [21]. The system includes eigenvalues and eigenvectors extraction as face encodings to be classified using kNN in the final classifier. Another method of a lazy learning algorithm is the lazy naïve-bayesian classifier. There are several variations of the naïve-Bayes algorithm. One of the most popular is the GNB. GNB differs slightly from regular naïve-Bayes in that GNB uses Gaussian data distribution for the estimation. The previous research uses GNB to determine the presence of cancer and produces accuracy as high as 98% in detecting breast cancer [22].

III. Research Method

A. Proposed Work

The research uses face images to build both race classifier and face recognition models. The whole pipeline of the architecture is represented in Fig. 1. The research performs the preprocessing phase from the input image, which consists of face alignment and face cropping. After the preprocessing phase, the image is inputted into a race classifier to determine which race the image belongs.

In the experiment, the researchers use three broadly generalized racial groups: black, Asian, and white. The face region is extracted from the selected frames using face alignment and face cropping. Then, the preprocessed face is inputted into the race classifier. The predicted race determines which encoder to extract the face data. The final classifier is a lazy learning classifier.

Each race group is assigned to a different face recognition agent that functions as a face encoder to produce a face encoding from the input image. The race prediction also determines which database it will be estimated with. For example, if an input image is classified as black, it will not be compared to the database that contains white people. However, it will be compared to the database that contains black people only. This operation will result in higher accuracy because the search space is smaller, and the face recognition systems are more focused. Each face recognition system uses a database that is composed of people with similar ethnic backgrounds, so the generalization of data will be minimized.

B. Dataset

The dataset is partitioned into two parts. The first part is the dataset for developing the race classifier. This first dataset uses the CelebA dataset [24]. The

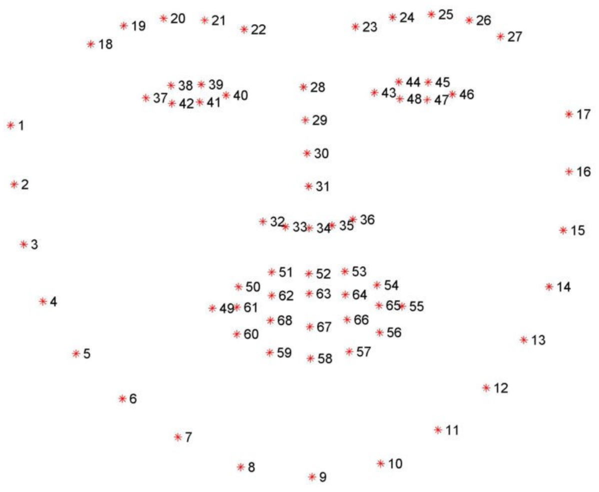

Fig. 2. Visualization of the 68 facial landmark coordinates from the iBUG 300-W dataset [23].

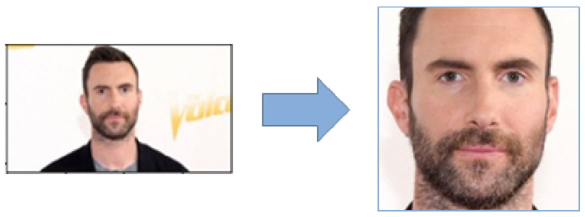

Fig. 3. Face detection and alignment result. Face alignment reduces the unnecessary variation impact of tilting face images.

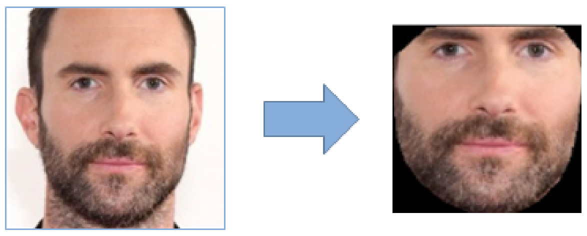

Fig. 4. Face cropping result. Face cropping reduces the noises present in the picture.

dataset contains more than 200,000 images. The training phase uses 10,000 manually race-tagged images out of the CelebA dataset. From the 10,000 manually tagged images, further balancing is applied according to the class with the lowest number of data. From the balancing process, around 2,200 images from each class (Asian, black, and white) are used for the training phase. Then, an additional 1,200 manually searched face images from Google Images are added for testing purpose.

The second partition of the dataset is used to develop the face recognition agents. The dataset consists of 2,000 manually collected images of famous people from Google Images with 100 different identities. The second dataset has an imbalance in the racial representation of the data to mimic the imbalances in mainstream datasets. The ratio of the dataset used for face recognition for white: Asian: black is 5: 3: 2. It means that different face recognition agents will learn from different amounts of data in the training phase.

C. *Dataset Preprocessing*

Every image in the dataset is preprocessed. The preprocessing phase intends to prepare the dataset so that the system can minimize the noise that appears on the image, like background objects, face orientation, and others. Using the pre-trained model by [23], the system tries to detect the facial landmarks present in the image. The pre-trained model is trained to identify 68 coordinates that map facial structures on the face. The indexes of the 68 coordinates are visualized in Fig. 2. The 68 coordinates are used to transform the image so the image will be aligned to be in the upright position. It is done to generalize the position of facial landmarks like nose, eyes, mouth, and others.

The following preprocessing phase is face cropping. Aligned faces produced by the face alignment phase still contain lots of noises. Face cropping is applied to the aligned faces to eliminate these noises. Face cropping is done by making a mask in the form of a filled convex polygon from the outer ring of the 68 landmarks. The area outside the created polygon is removed from any value. The preprocessing phase is visualized in Figs. 3 and 4. The preprocessing phase saves the image into 224 $\times$ 224 pixels to uniform it into one size. This size is chosen because the used networks do not use images bigger than 224 $\times$ 224 pixels.

TABLE I
SIMPLE CONVOLUTIONAL NEURAL NETWORK (CNN) RACE CLASSIFIER.

Layer	Model
1	Conv. 3 × 3, 64
2	MaxPool 2 × 2
3	Conv. 3 × 3, 64
4	MaxPool 2 × 2
5	Conv. 3 × 3, 32
6	Fully Connected (FC), 64, ReLU
7	Fully Connected (FC), 3, softmax

D. Race Classifier

The race classifier is trained using labeled image data. The data are classified into three groups of broadly generalized races: black, Asian, and white. There are several approaches to achieve the race classifier that performs well. In training the race classifier, the research uses categorical cross-entropy (Eq. (1)) as the loss function for approaches. In Eq. (1), y refers to a vector that determines the target class of the given data. Then, h is the prediction that the model makes to the corresponding data. Meanwhile, i refers to the iteration number, and N is the total number of iteration or data.

$$CCE(y,h) = -\sum_{i}^{N}(y_i \log h_i) \tag{1}$$

The first approach is to straight-up create a CNN network that consists of seven layers and train it from scratch. Table I shows the base model for the experiment architecture. There are several other variations that are tried with slight changes to the number of applied filters to each layer. However, this configuration achieves the best result among other variations. This approach is also made to compare the result to the achieved result using transfer learning on pre-trained networks.

The second approach uses transfer learning on Xception [17]. Xception is a CNN architecture published in 2017. Xception gets its name from “Extreme Inception”. Inception is a CNN architecture that uses depthwise separable convolution and skips connections. Trained using the ImageNet dataset, Xception achieves 79.81% for its top-1 accuracy. The transfer learning is done by replacing the classifier block of Xception with a new classifier with three output nodes because of the three available target groups. The new classifier block includes a dropout layer to avoid possible overfit with the network.

The third approach applies transfer learning on NASNet-A [25]. NASNet-A is a product of Network Architecture Search (NAS). It means that NASNet-A is not designed by humans but by a designed network to build a classifier network. NASNet-A achieves 82.7% on top-1 accuracy using the ImageNet dataset. The procedure of transfer learning applied to NASNet-A is the same as the procedure for Xception. Table II can also be referenced to look up the model of the new classifier block to replace the original classifier block of NASNet-A.

TABLE II
REPLACEMENT CLASSIFIER BLOCK FOR XCEPTION.

Layer	Model
1	Fully Connected (FC) 1024, ReLU
2	Dropout 0.5
3	Fully Connected (FC) 3, softmax

TABLE III
REPLACEMENT CLASSIFIER BLOCK FOR VGG16.

Layer	Model
1	Flatten
2	Fully Connected (FC) 1000, ReLU
3	Fully Connected (FC) 1000, ReLU
4	Dropout 0.5
5	Fully Connected (FC) 3, softmax

The fourth and final approach is transfer learning on VGG16 [16]. VGG16 is a CNN-based architecture that consists of 16 layers, so its name is VGG16. VGG16, published in 2014, achieves 74.4% on its top-1 accuracy using the ImageNet dataset.

Transfer learning on VGG16 also involves changing its original classifier block with a three-node output to accommodate the necessity. The new classifier block on VGG16 is shown in Table III. Each classifier that is attained by performing transfer learning is experimented with using both classifier blocks in Tables II and III. It is only the case that Xception and NASNet-A achieve better results using the classifier described in Table II, and VGG16 has better results using the classifier described in Table III.

Each network for transfer learning is only allowed to change the weights of the last convolution block right before the custom classifier block. It is to preserve the pre-trained weights achieved from the original dataset. It freezes all the layers except the last convolutional block and the last classifier block.

$$f(x_i) = \frac{e^{-x_i}}{\sum_j^N e^{-x_j}} \tag{2}$$

Equation (2) is the softmax activation function [26]. Each classifier is ended with a softmax activation function since it is a classifier network. The softmax activation function is used to accommodate the use of categorical cross-entropy to classify the input into more than two groups. The best performing race classifier is

TABLE IV
21 LAYERS IN CONVOLUTIONAL AUTO-ENCODER ARCHITECTURE.

Layer	Model	Layer	Model
1	Conv. 3 × 3, 128	11	Conv. 3 × 3, 8
2	MaxPool 2 × 2	12	UpSampling 2 × 2
3	Conv. 3 × 3, 64	13	Conv. 3 × 3, 16
4	MaxPool 2 × 2	14	UpSampling 2 × 2
5	Conv. 3 × 3, 32	15	Conv. 3 × 3, 32
6	MaxPool 2 × 2	16	UpSampling 2 × 2
7	Conv. 3 × 3, 16	17	Conv. 3 × 3, 64
8	MaxPool 2 × 2	18	UpSampling 2 × 2
9	Conv. 3 × 3, 8	19	Conv. 3 × 3, 128
10	MaxPool 2 × 2	20	UpSampling 2 × 2
	-	21	Conv. 3 × 3, 1

then used in the final architecture to perform the best available race classification. The performance of each network is measured with its accuracy.

E. Face Encoders

Face encoders are the part of the network to extract features from input images. In this experiment, there are two types of networks: auto-encoder and CNN. All networks are experimented with and trained three times for each variation because each race group is supposed to have its face encoder models.

Auto-encoders are an unsupervised learning neural network with a backpropagation process. This algorithm does not need data labels. Auto-encoders score their performance by comparing the input image with the reconstructed image. Auto-encoder consists of two parts: the encoder part and the decoder part. The encoder part will reduce the data dimension, and a smaller representation of the input called the encoding is attained. Moreover, the decoder part will try to reconstruct the encoding back into the original input. In conclusion, the idea of auto-encoders is to reduce the original data dimension without losing essential details. The encoding quality is scored by how well it can be reconstructed back into the original data.

After the training phase, the auto-encoder can be integrated with a lazy learning algorithm to classify the encodings. The system only needs the encoder part of the auto-encoder to perform the reduction on the input data to do this. The decoder part of the auto-encoder is only used for the training phase. The size of the encoding is more important and impactful than the configuration of the auto-encoder itself [13]. With this concept in mind, the researchers do several experiments with varying encoding sizes and configurations on the convolutional layers on the CAE. The researchers experiment with architectures using 21 layers, 17 layers, 15 layers, 13 layers, and 9 layers.

The biggest and the smallest network are shown in Tables IV and V. The convolutional layers on the CAEs are equipped with Rectified Linear Unit (ReLU) [27] (Eq. (3)) as the activation function. The researchers also experiment with using sigmoid and softmax, but the results are inferior to the result achieved by using ReLU.

TABLE V
9 LAYERS IN CONVOLUTIONAL AUTO-ENCODER ARCHITECTURE.

Layer	Model	Layer	Model
1	Conv. 3 × 3, 128	5	Conv. 3 × 3, 32
2	MaxPool 2 × 2	6	UpSampling 2 × 2
3	Conv. 3 × 3, 32	7	Conv. 3 × 3, 128
4	MaxPool 2 × 2	8	UpSampling 2 × 2
	-	9	Conv. 3 × 3, 1

$$f(x) = \max(0, x) \tag{3}$$

$$\mathrm{SSIM}(h, y) = \frac{(2\mu_h\mu_y + (k_1L)^2)(2\sigma_{hy} + (k_2L)^2)}{(\mu_{h^2} + \mu_{y^2}(k_1L)^2)(\sigma_{h^2} + \sigma_{y^2}(k_2L)^2)} \tag{4}$$

To calculate the reconstruction score, the researchers use the Structural Similarity Index (SSIM) [28] as the loss function for the face auto-encoder. Equation (4) is the formulation for SSIM. SSIM is used to measure how similar the target image (h) and output image (y) are. In Eq. (4), μ_h refers to the average of h, and μ_y is the average of y. Then, σ_h^2 is the variance of h, and σ_y^2 refers to the variance of y. Meanwhile, $(k_1L)^2$ and $(k_2L)^2$ are two variables to stabilize the division with a weak denominator. The value of k_1 is 0.001, the value of k_2 is 0.003, and L is a dynamic range of the pixel values.

Another experiment to develop a face encoder uses FaceNet [29]. FaceNet is a face encoder that is trained using the triplet loss function. The triplet loss function calculates the loss using three inputs: anchor, positive, and negative. The anchor is a different image with the same identity of the image, which is represented as positive. However, negative is an image of a different identity to the represented identity by the anchor. Triplet loss intends to close the gap between the encoded values of anchor and positive and widens the gap between the encoded values of anchor and negative. The formula for triplet loss is shown in Eq. (5).

$$\begin{aligned} L(A, P, N) = \max((f(A) - f(P))^2 \\ - (f(A) - f(N))^2 + \alpha, 0) \end{aligned} \tag{5}$$

In Eq. (5), the notation of A refers to the anchor image, and P is the positive image that contains a different picture of the same person with the anchor. Then, N refers to a negative image that contains an image of a different person from the anchor and the positive. The researchers use α as a margin to add the

TABLE VI
ADDITIONAL CLASSIFIER BLOCK FOR INCEPTION-RESNET-V1.

Layer	Model
1	GlobalAveragePooling
2	Dropout 0.2
3	Fully Connected (FC) n, softmax

TABLE VII
REPLACEMENT CLASSIFIER BLOCK FOR VGGFACE.

Layer	Model
1	Fully Connected (FC) 100, tanh
2	Dropout 0.3
3	Fully Connected (FC) 10, tanh
4	Dropout 0.2
5	Fully Connected (FC) n, softmax

minimum distance between A-P and A-N to maximize the gap. The loss function chooses the maximum value between 0 and the calculated distance. If the calculated distance is less than 0, the network fails to calculate the distance between A-P that is close enough, or the distance between A-N is far enough. The distance can be calculated using the standard multidimensional Euclidean distance (Eq. (6)). The d is the distance between objects of p and q. Then, n refers to the number of dimensions the data are arranged of, and i is an iterative variable to determine which dimension value is processed.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{6}$$

In the original research [29], FaceNet is released with two architectures. The first architecture is GoogLeNet [30] and the second architecture is Inception-ResNet-v1 [31]. The transfer learning applied to FaceNet architectures using the triplet loss function can be done by adding a dataset in the training phase. However, the researchers decide only to perform transfer learning using the triplet loss function to GoogLeNet.

The transfer learning performed on Inception-ResNet-v1 uses the categorical cross-entropy loss function by adding a classifier block to the network. It compares the result between networks that have been trained with the same dataset but re-trained using different ways. The final layer of the Inception-ResNet-v1 contains various configurations regarding the number of the nodes because, for each race, the number of the available class is different. The only parts of the networks that are allowed to change their weights are the last convolutional blocks.

For the Inception-ResNet-v1, the additional classifier block is also allowed to change weights. Table VI shows the additional classifier block for Inception-ResNet-v1. This rule is enforced by freezing the other layer and intends to preserve the pre-trained identity. Hence, the network does not lose the pre-trained aspect.

Another experiment is the transfer learning procedure using VGGFace [32]. VGGFace is a pre-trained VGG16 network that is trained with a dataset named VGGFace. VGGFace dataset contains 2,622 different identity labels with over 2.6 million images. The architectural difference between VGG16 and VGGFace is located on the classifier block because of the different available categories in the VGGFace dataset and the ImageNet dataset. The transfer learning procedure is done by giving VGGFace a new classifier block. The new classifier block is shown in Table VII. The only part of the original network that is allowed to change the weights is the last convolution block. The number of the nodes of the last fully connected layer in the classifier block is also dependent on which race group it is trained on. Then, the best-performing face encoder will be used in the main system to encode faces according to the predicted race that it is assigned upon.

F. Lazy Learning

Lazy learning is used as the final classifier to make the system more flexible to a less stable data flow. Using a lazy learning algorithm, the system can better adapt in the case of additional data that the system has never known in the training phase. At the lazy learning stage, the used database only contains identities from the same race group, resulting from the race classifier. The researchers experiment with two lazy learning algorithms, which are kNN and GNB.

The kNN works by looking for the k-nearest neighbor and predicting the class of the input based On the majority class of the nearest neighbors, the input that the kNN classifier will receive is the encoding as the product of the face encoder. Distance between face encodings is calculated using Euclidean distance (Eq. (6)).

GNB is a probabilistic classifier. Basic naïve-Bayes can only calculate absolute values and cannot process the continuous values. When dealing with continuous values, the data can be distributed according to Gaussian distribution. The Gaussian or normal distribution is mapped using the probability density function shown in Eq. (7). The μ refers to the mean of the distribution. Meanwhile, σ shows its standard deviation, and σ^2 refers to the variance of the distribution. Then, using the Gaussian distribution, continuous data can be calculated using the regular naïve-Bayes formulation

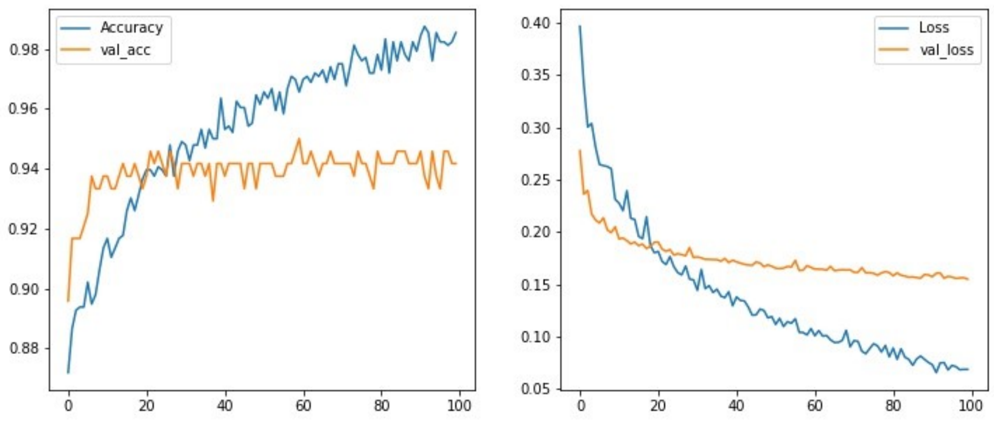

Fig. 5. Best training performance for race classification using VGG16 transfer learning result. The figure shows the best-performing experiment. The legends of “val_acc” and “val_loss” in the figure refer to the accuracy and the loss value of the validation dataset. The X-axis is the iteration number, and the Y-axis shows the value of accuracy and loss in their respective graphs.

in Eq. (8). The $p(C_z|x)$ refers to a probability of a value represented by z for a condition represented by C towards a target class represented by x.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{7}$$

$$p(C_z|x) = \frac{p(C_z)p(x|C_z)}{p(x)} \tag{8}$$

IV. Results and Discussion

The experiment is conducted in two major parts. The first part consists of developing a race classifier, and the second part is for developing a face encoder. The performance of the face encoders is measured using the face recognition rate from the lazy learning classifiers.

A. Race Classifier

The training phase uses 6,600 face images in the dataset. Meanwhile, the testing phase consists of 1,200 images from the testing dataset. The best-achieved accuracy from the developed race classifier is 98.44% from the transfer learning procedure of VGG16 (see Table VIII). There are four architectures that the research experiments with. The architectures are simple CNN, Xception, NASNet-A, and VGG16. Figure 5 only shows the performance of the training and the validation phase of the best-performing architecture, which is VGG16. The experiment with the best result uses VGG16. VGG16 is experimented with using SGD and Adam optimizers along with a learning rate of 0.001 and 0.0001. Then, the training is done with 100 epochs because VGG16 shows steady progress on its accuracy and loss. The model attains 95% accuracy on the validation set and an even better 98.45% on the testing set.

TABLE VIII
Race Classifier Results.

Model	Test Accuracy
7-Layer Simple CNN	44.16%
Xception (TL)	36.66%
NASNet-A (TL)	42.50%
VGG16 (TL)	98.45%

TL: Transfer Learning

The comparison of best performances from the conducted experiments is shown in Table VIII. The experiment concludes that VGG16 performs the best among other models. The experiment with simple CNN is done by training the network from scratch. The experiment has 50 epochs. After 10 epochs, none of the models continue to improve their accuracy. Moreover, the best accuracy is achieved by the model shown in Table VIII with an accuracy of 44.16% for the experiment using simple CNN as a race classifier when tested with the testing dataset.

Moreover, the transfer learning using Xception is done twice using both Adam optimizer [33] and SGD optimizer [34]. The variations of learning rate are 0.001 and 0.0001. There are four experiments done to apply the transfer learning procedure. Each experiment is done with 50 epochs. There is no significant improvement in each epoch of the training phase on the validation set. The best Xception model is achieved using Adam optimizer and a learning rate of 0.001. It has 87.05% accuracy on the training phase but only attains 36.66% on the testing set.

TABLE IX
CONVOLUTIONAL AUTO-ENCODER RESULTS WITH LAZY LEARNING.

Model (Encoded Dimension)	SSIM Loss	kNN Acc.	GNB Acc.
21-Layer (4 × 4 × 8)	0.2711	0.4167	0.3417
17-Layer (8 × 8 × 16)	0.1395	0.5000	0.4500
15-Layer (16 × 16 × 16)	0.0567	0.4500	0.3250
13-Layer (16 × 16 × 32)	0.0329	0.4500	0.4833
9-Layer (32 × 32 × 32)	0.0052	0.3417	0.4917

*All results recorded are averages of the three races.

Next, the transfer learning using NASNet-A has also been experimented with using both Adam and SGD optimizers. The learning rate is also experimented with the values of 0.001 and 0.0001. The experiment with NASNet-A uses 50 epochs and yields the best training result using Adam optimizer with a learning rate of 0.001 with a training accuracy of 89.09%. Although it has a high training accuracy, when it is tested with the testing set, the network only attains 42.5% accuracy.

B. Face Encoders

The training phase for face encoders uses 2,000 face images in the dataset with 100 different identities. The ratio for white: Asian: black is 5: 3: 2. Each identity in the database consists of 20 face images with those particular identities. Then, the testing phase has 120 images from the testing dataset. The performance of the face encoder models is measured using kNN and GNB. The best performing model is the model acquired through transfer learning using VGGFace with a face recognition rate of 85% using kNN. The face recognition rate when the model is given a new database containing five new identities on the white database, three new identities on the Asian database, and two new identities on the black database is 81.51%. These new identities have never met in the training phase, which rigid models like deep learning classifiers can not adapt to recognize.

The first experiment applies an auto-encoder to try to extract face encodings. There are five models with high similarity in the models' configurations. Among the used models (Table IX), the best-performing model uses the configuration, as shown in Table V (9-layer model). The model that produces the largest encoding size performs the best among other configurations [13]. The results of the auto-encoder with kNN and GNB for face recognition are shown in Table IX. From Table IX, it is concluded that the ability of reconstruction of an auto-encoder does not correlate with an ability to extract features that can be used by kNN and GNB classifiers.

The second experiment uses transfer learning with Inception-ResNet-v1 with pre-trained weights that are provided by FaceNet. The network is trained using the same method as the transfer learning of a deep learning classifier. After the training phase, the classifier block is removed from the network so the network can perform image encoding to be used along with lazy learning algorithms. The classifier block used is the one shown in Table VI. The experiment is done in two variations.

TABLE X
TRANSFER LEARNING RESULT WITH INCEPTION-RESNET-V1.

Trainable Layers	Race/ Class	Training Acc.	kNN Acc.	GNB Acc.
4 Layers	Black	0.5100	0.450	0.250
	Asian	0.3000	0.325	0.150
	White	0.2760	0.500	0.125
6 Layers	Black	0.5400	0.425	0.350
	Asian	0.3667	0.250	0.125
	White	0.2960	0.500	0.100

TABLE XI
TRANSFER LEARNING RESULT WITH GOOGLENET.

Race/ Class	kNN Acc.	GNB Acc.
Black	0.175	0.125
Asian	0.075	0.100
White	0.025	0.050

The first variation unfreezes four layers before the classifier block, and the second variation unfreezes 6 layers before the classifier block to change their weights. The result is shown in Table X. The particular model does not produce a good enough face encoder. From Table X, this particular model has the most trouble handling Asian face images.

The second FaceNet network for transfer learning applies the triplet loss function. In this experiment, GoogLeNet is used as the architecture and pairs up every available dataset to create sets of anchor, positive, and negative. Each set is used as a 1-time input to the network.

Only the last two convolutional blocks are unfrozen to change weights according to the training session. This experiment trains each model for 30 epochs. The transfer learning process does not yield any particularly good results. When tested using the testing set, none of the models achieves more than 20% of the face recognition rate using kNN or GNB. The testing result is shown in Table XI.

The final face encoder experiment involves transfer learning to VGGFace. VGGFace is given a new classifier block to accommodate each class's available names. The used classifier block is shown in Table VII. The experiment is done three times. The first one allows two last layers before the classifier block and the whole classifier block to be trainable. The second

TABLE XII
TRANSFER LEARNING RESULT WITH VGGFACE.

Trainable Layers	Race/ Class	kNN Acc.	GNB Acc.	Avg. of kNN Acc.	Avg. of GNB Acc.
2 Layers	Black	0.8500	0.8000	0.8167	0.73330
	Asian	0.8000	0.6750		
	White	0.8000	0.7250		
3 Layers	Black	0.8500	0.8250	0.8330	0.75000
	Asian	0.8500	0.7000		
	White	0.8000	0.7250		
4 Layers	Black	0.8500	0.8250	0.8500	0.75833
	Asian	0.9000	0.7250		
	White	0.8000	0.7250		
4 Layers	Single-Pipeline	0.8083	0.8083	0.8083	0.80830

one allows three last layers before the classifier block alongside the classifier block to be unfrozen. Meanwhile, the last one allows four last layers before the classifier block, along with the classifier block to be unfrozen.

The best result is from four trainable layers model with the system using three face encoders. It performs better than a single face encoder model when using kNN. The result is shown in Table XII. From the experiment, using four trainable layers yields the best result.

Other than the transfer learning to accommodate each race, the researchers also do an experiment that the researchers do not partition the database. The last row in Table XII describes the result of a no-race network which means only one model works to recognize three databases without using race discrimination. The result is better when using kNN for four trainable layers network than the model that uses single-pipeline. The network that uses a multiple-agent concept achieves 85% of face recognition rate. Meanwhile, the single-pipeline model only achieves 80.83% of the face recognition rate.

The best-performing system is tested against a new stream of data with an added amount of data. VGGFace, as the best performing system, still achieves high results even though it faces new data with a completely new identity. The result of this test is shown in Table XIII. Using the new data, the system using VGGFace can still maintain decent accuracies with both kNN and GNB. The test is executed by giving two new queries for each new identity in the database. The database is also repopulated with the new data so the lazy learning algorithm can do a re-estimation. Like the previous identities, the new identities also consist of 20 face images of that particular identity. From the face encoder experiments, VGGFace is the only transfer learning to yield decent results. It may be caused by the quality of the new dataset or the procedure of transfer learning that is not the most effective on other architectures.

TABLE XIII
TEST RESULT WITH NEW DATA.

Race/ Class	N of New Subject	kNN Acc.	GNB Acc.	kNN with New Data	GNB with New Data
Black	2	0.8409	0.7727	10.000	0.75
Asian	3	0.8043	0.8043	0.6666	0.66
White	5	0.8000	0.7800	0.8000	0.70

C. Lazy Learning

The two algorithms for the final classifier are kNN and GNB. Throughout the experiments, different configurations of k in kNN have been tried. It is found that using seven nearest neighbors generally results in the best performances. However, GNB generally performs slightly worse than kNN in the experiments. Testing the GNB module takes less time than kNN because GNB does not have any parameters to be tested and tuned. Usage of kNN also has better results when facing new data with new identities.

D. Promising Potential

The researchers are optimistic that the research can be continued and developed with various configurations. The researchers would like to keep gathering images to increase the accuracy of the models in this system. The addition to the dataset can be done in multiple ways. The researchers can increase the number of images in each identity. Currently, the system employs a database that contains 20 face images for each identity.

Table XIV shows some predictions made along with the image queries and the person's real identity in the query image. The model still has a bit of a struggle in distinguishing faces with similar head shapes. The researchers believe that by adding more population into each identity, the system can employ more than one algorithm as the end classifier uses an ensemble of lazy learning. Each part of the ensemble can use different parts of the dataset on each face. Therefore, the confidence of the network can be boosted to even higher levels.

Another way to add data is by adding the classes, in this case, another amount of identities. Adding more identities to each dataset according to each race group may improve the accuracy of individual face recognition models. Looking into Table XII, the researchers can see that the 4-layer trainable model achieves the best performance with GNB in the recognition task

TABLE XIV
TRANSFER LEARNING RESULTS WITH VGGFACE USING K-NEAREST NEIGHBOR.

Image	Real Identity	Prediction
	Bill Clinton	Bill Clinton
	Bon Jovi	Bon Jovi
	Octavia Spencer	Oprah Winfrey
	Kevin Hart	Kevin Hart
	Halle Berry	Halle Berry
	Andy Lau	Andy Lau
	Sakura Miyawaki	Bae Joohyun
	David Beckham	David Beckham
	Whoopi Goldberg	Whoopi Goldberg
	Jo Insung	Jo Insung
	Hirai Momo	Hirai Momo
	Jackie Chan	Jackie Chan

for the white race group. The different amount of variation to the data is suspected to correlate with different accuracies. The current number of identities of the white race group is 50 people. Meanwhile, the current number of identities of the Asian race group is 30 people. Then, the current number of identities of the black race group is 20 people. Adding the amount of variation in the categories or identities may improve the capability of the face encoders to discriminate each face image further. Hence, the most defining features of faces can be extracted. The researchers do not need to worry about the disparity in the amount of data in each race because each race group is covered by and assigned to its face recognition model.

V. Conclusion

The research introduces a method by reducing racial bias in face recognition using a multiple-agent architecture based on racial features. The research is only done using 100 identities comprised of three distinct racial groups. Then, each used picture is in a condition of uniform lighting. The multiple deep learning agents achieve 85% accuracy, while the single-pipeline model only achieves 80.83% accuracy with the same data. This method can be applied to any number of racial groups according to the dataset composition. However, the research discovers that by itself, CAE extractions are not good enough to be classified as far as face identification goes. The lazy learning algorithms do not recognize any similarity in the distribution data of the encoded images of the same identity. For the feature extraction, it is proven that transfer learning using deep learning methods works well when the classifying blocks are removed.

The researchers intend to explore other possibilities to improve our multi-agent system further. Further experiments will try to include more variations of races and faces. Architectural modification can also be considered better to suit a more specific need for the system.

References

[1] R. S. Malpass and J. Kravitz, "Recognition for faces of own and other race," *Journal of Personality and Social Psychology*, vol. 13, no. 4, pp. 330–334, 1969.

[2] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. PMLR, 2018, pp. 77–91.

[3] J. Wang and Z. Li, "Research on face recognition based on CNN," in *IOP Conference Series: Earth and Environmental Science*, vol. 170. IOP Publishing, 2018, pp. 1–5.

[4] T. Rajeshkumar, U. Samsudeen, U. Scholar, S. Sangeetha, and U. S. Rani, "Enhanced visual attendance system by face recognition using K–Nearest Neighbor algorithm," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 11, no. 06-Special Issue, pp. 141–147, 2019.

[5] Y. Zhang, E. Zhang, and W. Chen, "Deep neural network for halftone image classification based on sparse auto-encoder," *Engineering Applications of Artificial Intelligence*, vol. 50, pp. 245–255, 2016.

[6] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "HCP: A flexible CNN framework for multi-label image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1901–1907, 2016.

[7] O. Déniz, G. Bueno, J. Salido, and F. De La Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598–1603, 2011.

[8] B. H. Park, S. Y. Oh, and I. J. Kim, "Face alignment using a deep neural network with local feature learning and recurrent regression," *Expert Systems with Applications*, vol. 89, pp. 66–80, 2017.

[9] T. Vo, T. Nguyen, and C. T. Le, "Race recognition using deep convolutional neural networks," *Symmetry*, vol. 10, no. 11, p. 564, 2018.

[10] W. Wang, F. He, and Q. Zhao, "Facial ethnicity classification with deep convolutional neural networks," in *Chinese Conference on Biometric Recognition*. Chengdu, China: Springer, Oct. 14–16, 2016, pp. 176–185.

[11] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations*. MIT Press, 1987, pp. 318–362.

[12] S. Gao, Y. Zhang, K. Jia, J. Lu, and Y. Zhang, "Single sample face recognition via learning deep supervised autoencoders," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2108–2118, 2015.

[13] I. Manakov, M. Rohm, and V. Tresp, "Walking the tightrope: An investigation of the convolutional autoencoder bottleneck," *ArXiv Preprint ArXiv:1911.07460*, 2019.

[14] A. Tewari, M. Zollhofer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt,

“MoFA: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 1274–1283.

[15] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *International Conference on Artificial Neural Networks*. Rhodes, Greece: Springer, Oct. 4–7, 2018, pp. 270–279.

[16] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 7–9, 2015.

[17] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[18] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[19] H. C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.

[20] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[21] J. P. Jose, P. Poornima, and K. M. Kumar, “A novel method for color face recognition using KNN classifier,” in *2012 International Conference on Computing, Communication and Applications*. Dindigul, India: IEEE, Feb. 2012, pp. 1–3.

[22] H. Kamel, D. Abdulah, and J. M. Al-Tuwaijari, “Cancer classification using Gaussian Naive Bayes Algorithm,” in *2019 International Engineering Conference (IEC)*. Erbil, Iraq: IEEE, June 23–25, 2019, pp. 165–170.

[23] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, “300 faces in-the-wild challenge: The first facial landmark localization challenge,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013, pp. 397–403.

[24] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[25] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[26] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *ArXiv Preprint ArXiv:1811.03378*, 2018.

[27] A. F. Agarap, “Deep learning using Rectified Linear Units (ReLU),” *ArXiv Preprint ArXiv:1803.08375*, 2018.

[28] J. Nilsson and T. Akenine-Möller, “Understanding SSIM,” *ArXiv Preprint ArXiv:2006.13846*, 2020.

[29] 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.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017, pp. 4278–4284.

[31] 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.

[32] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in *BMVC*. British Machine Vision Association, 2015, pp. 1–12.

[33] D. P. Kingama and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference for Learning Representations*, San Diego, USA, May 7–9 2014.

[34] S. Ruder, “An overview of gradient descent optimization algorithms,” *ArXiv Preprint ArXiv:1609.04747*, 2016.