IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 1, 2020, accepted July 9, 2020, date of publication July 17, 2020, date of current version July 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010018

Pyramid With Super Resolution for In-The-Wild Facial Expression Recognition

THANH-HUNG VO, GUEE-SANG LEE, (Member, IEEE), HYUNG-JEONG YANG, (Member, IEEE), AND SOO-HYUNG KIM, (Member, IEEE)
Department of Electronic Computer Engineering, Chonnam National University, Gwangju 61186, South Korea

Corresponding author: Soo-Hyung Kim (shkim@jnu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Ministry of Education through the Basic Science Research Program under Grant NRF-2018R1D1A3A03000947 and Grant NRF-2020R1A4A1019191.

ABSTRACT Facial Expression Recognition (FER) is a challenging task that improves natural human-computer interaction. This paper focuses on automatic FER on a single in-the-wild (ITW) image. ITW images suffer real problems of pose, direction, and input resolution. In this study, we propose a pyramid with super-resolution (PSR) network architecture to solve the ITW FER task. We also introduce a prior distribution label smoothing (PDLS) loss function that applies the additional prior knowledge of the confusion about each expression in the FER task. Experiments on the three most popular ITW FER datasets showed that our approach outperforms all the state-of-the-art methods.

INDEX TERMS Emotion recognition, image resolution, human computer interaction.

I. INTRODUCTION

Non-verbal communication plays an essential role in person-person communication. These non-verbal signals can add clues, additional information, and meaning to spoken (verbal) communication. Some studies estimate that around 60% to 80% of communication is non-verbal [1]. These signals include facial expressions, eye contact, voice tone and pitch, gestures, and physical distance, of which facial expression is the most popular input for analysis. The facial expression recognition (FER) task aims to recognize the emotion from the facial image.

In psychology and computer vision, emotion can be divided into two kinds of model: discrete and dimensional continuous [2]–[4]. The dimensional continuous model focuses on arousal and valence, which values from -1.0 to 1.0, whereas the discrete emotion theory classifies a few core emotions such as happy, sad, angry, neutral, surprise, disgust, fear and contempt. In our study, we attempted *discrete emotion recognition*.

Ekman and Friesen developed a Facial Action Coding System (FACS) to analyze human facial movements [5]. However, this scheme needs trained humans and is extensively time consuming. The recent advances of successful machine learning in computer vision could help simplify and automate those processes. The scope of our study is automatic facial expression recognition, where emotional expression is in the discrete model.

Many studies use traditional image processing and machine learning for the FER task. Shan *et al.* used local statistical features, termed Local Binary Patterns, for person-independent facial expression recognition [6]. Ma and Khorasani used one-hidden-layer feed forward neural network on a two-dimensional discrete cosine transform [7]. Lien *et al.* combined facial feature point tracking, dense flow tracking, and gradient component detection to detect FACS and calculate emotion [8]. In [9], Zhang *et al.*extracted scale-invariant feature transform and used the deep neural network (DNN) as the classifier. Aleksic and Katsaggelos used hidden Markov models for automatic FER [10].

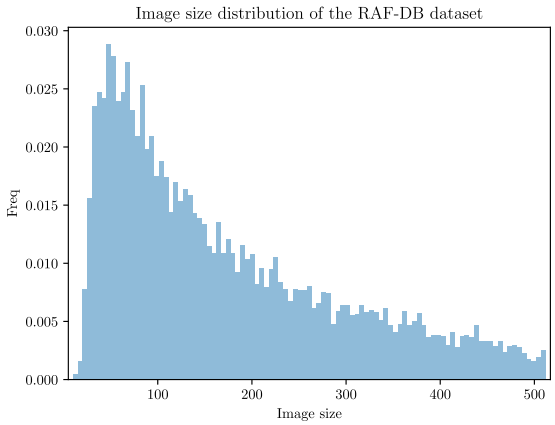
Recently, deep learning (DL) has significantly affected many fields, such as image, voice, and natural language processing. In the Boosted Deep Belief Network [14] introduced by Liu *et al.*, multiple deep belief networks learned feature representation from patches of image and some of them were selected to boost. In [15], Liu *et al.* ensembled three convolutional neural networks (CNN) subnets and concatenated the outputs to predict the final results. Huang [16] used a custom residual block of the ResNet architecture

The associate editor coordinating the review of this manuscript and approving it for publication was Waleed Alsabhan.

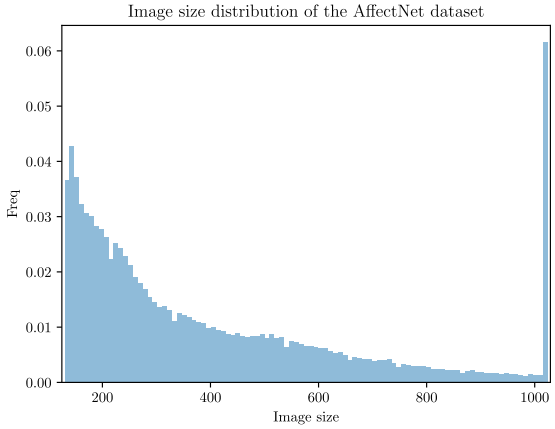
and late fusion to combine the results from the VGG and the ResNet models. Zeng *et al.* extracted image histogram of oriented gradients and passed them through deep sparse autoencoders to classify them [17]. Tozadore *et al.* grouped emotions into several groups to help CNN classify with better accuracy [18].

Despite these successes of in-the-lab datasets, the rise of the in-the-wild (ITW) dataset in recent years has raised new challenges for researchers. When in-the-lab datasets were collected under control, the data were clean, accurate, and uniform. In contrast, ITW datasets are noisy, inaccurate, and variant. We outline the following two observations about ITW datasets for the FER task.

Observation 1: **The images size of the ITW datasets varies**. While the size of in-the-lab datasets images is controlled and nearly constant, ITW dataset images have various sizes from too small to large. Figure 1 shows the image size distribution of the RAF-DB [11], [12] (Fig. 1a) and the AffectNet [13] dataset (Fig. 1b). These two selected datasets are the most popular ITW datasets for the FER task. Because of the differences in width and length, the average of the two is considered as the size of the image. In both datasets, the small images occur more frequently and this frequency



(a) RAF-DB

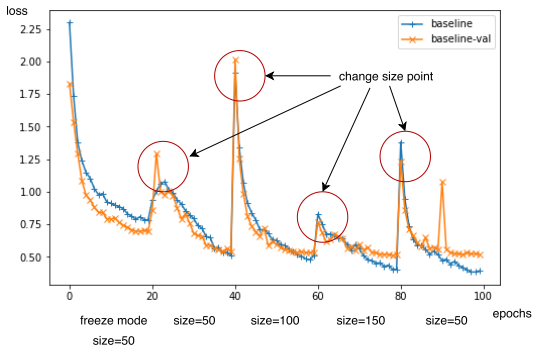


(b) AffectNet

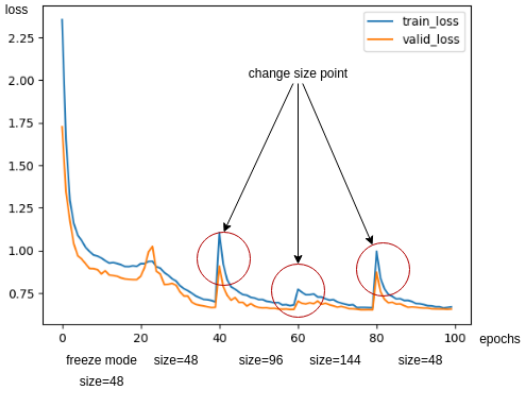
FIGURE 1. **The image size distribution of the RAF-DB [11], [12] and AffectNet [13] datasets.**

decreases with increasing size. The mean and variance of the image size in the RAF-DB are 193 and 144, which is a bit large. The AffectNet dataset has larger image sizes, ranging from 130 pixels to more than 2000 pixels. In the graph, we round all images larger than 2000 pixels to the fixed value of 1000 pixels. Similar to the RAD-DB dataset, the number of image decreases when the size of the image increases. The third most popular ITW datasets for the FER task is the FER+ dataset [19] extended from the FER2013 [20]. It also faces the different-image-size problem. Unfortunately, the original image size information was omitted when the author of the FER data published. Most of the studies in this field does not consider the image-size problem. They simply resized all images to the same size, e.g. 128 × 128 or 224 × 224. The first reason is due to the DL framework itself, because in the batch mode, each batch must have the same input shape. Implementing different input sizes at the same time takes more effort, and is complicated and computationally inefficient. While CNN architecture was successful for many image classification tasks, it is based on the assumption that despite the resizing of the images the network could learn to distinguish by itself. Nearest-neighbor interpolation, bilinear, and bicubic algorithms are popular techniques to scale image sizes.

Observation 2: **The CNNs are usually sensitized with input image size.** While CNN was very successful for many tasks related to image classification and segmentation, this architecture suffers from several weaknesses. One of CNN's weaknesses is the sensitivity to the size of the input image. Zooming is one of the data augmentation techniques that attempts to address this problem. The selected zooming scale in most of the experiments ranged from 0.9 to 1.2 because values outside this range degraded and damaged the network. With *global pooling*, CNN networks could support different input sizes, and the size incremental technique was used to train the networks more quickly and gives coverage easier. Despite the improvement offered by this process, the network remains sensitive to the input size. Therefore, the network trained with this input size works poorly with the same images but on a different scale. Figure 2 shows the training and validation loss for VGG16 when training with the RAF-DB and the FER+ in different scales: 50 × 50, 100 × 100, 150 × 150 and back to 50 × 50 again in RAF-DB and 48 × 48, 96 × 96, 192 × 192 and again 48 × 48 in the FER+ for every 20 epochs in the sequence. We use weights transfer from the ImageNet [21], and then, we freeze the whole CNN architecture except the fully connected layers. The freeze steps were trained in 20 epochs at the smallest input image size. At the point of image size change (epoch 41, 61, 81), the loss of both training and validation set a significant increase. At the epoch 81, although the input size returns to the size 48 × 48 that was used to train to the network before, the loss value still increases because of the characteristics of convolution. The convolution layer uses a kernel (size 3 × 3, 5 × 5, or similar) to scan the "pixel" in the previous layer. Then, even though the image is the same but in a different scale, the next convolution



(a) RAF-DB



(b) FER+

FIGURE 2. The loss value for the training and validation during the training process as the input size changed for the RAF-DB and the FER+ (VGG16 architecture [22]).

layers learns very different features; therefore, increasing the kernel size does not help here.

While currently, the super-resolution (SR) step was in the pre-processing for input, it could be a part of the DL architecture. SR approaches may be better than the older algorithms such as nearest-neighbor interpolation, bilinear, and bicubic to solve the small-image-size problem. The SR task is used to make the larger image from a low-resolution image while trying to fill the lost pixels and to avoid the pixels becoming blurred. From a low-resolution image, e.g. size $W \times H$, the SR task is used to make the larger image $kW \times kH$ where $k \geq 2$, with the aim of making the new image as clear as possible. While down-scaling the image from high-resolution to low-resolution is an easy task, the reverse direction is not. The missing pixels that are lost from low-resolution need to be recovered. Some recent research has focused on this problem. Dong *et al.* introduced the Super-Resolution Convolutional Neural Network (SRCNN), a deep CNN model that works on low-resolution and high-resolution feature maps and finally generated a high-resolution image [23]. The SRCNN is lightweight and outperforms the bicubic interpolation. Very Deep Super Resolution (VDSR) has a similar structure as the SRCNN but is more in-depth [24]. In [25], Shi *et al.* makes the efficient sub-pixel convolutional neural network (ESPCN) that outperforms the SRCNN. ESPCN improves SRCNN by dealing with the feature maps at low-resolution and upsampling to the final image. Ledig *et al.* used resblocks to build SRResNet in [26]. Lim *et al.* proposed enhanced deep super-resolution network (EDSR) [27]. The EDSR is a modified version of the SRResNet that removes all batch normalization layers to reduce computing by 40% while improving the efficiency. They also designed a multi-scale network from the base block with good results. Hu *et al.* published a Cascaded Multi-Scale Cross network that includes a sequence of the cascaded sub-networks [28]. In recent years, the network for the SR has deepened, and the accuracy has been improved more. While the SRCNN is lightweight but low accuracy, the EDSR needs more computing but generates better results.

Our study has two highlight contributions. Firstly, we propose a Pyramid with Super-Resolution (PSR) network architecture to deal with the different-image-size problem for the ITW FER task. Our approach aims to view an image on several scales and uses SR for up-scaling. With many small-size-image problems in real-world FER datasets, the SR improves the network performance. Viewing the image on many scales also helps the network to learn not only at a small local but also at the global view of input. We also discuss the loss function and apply it to the FER task where the distribution of confusion labels are known and can be used.

The rest of this paper is organized as follows. We explain our proposed methods in section II and introduce the prior distribution label smoothing (PDLS) loss function in section III. Dataset information is presented in section IV. Section V describes the experimental results and discussion. Finally, we conclude our study in section VI.

II. PYRAMID WITH SUPER-RESOLUTION (PSR) NETWORK

We deal with the various image-size problems by using a pyramid architecture, which is termed as the PSR network. Figure 3 shows the overall PSR network architecture. There are six blocks in our approach, including spatial transformer network (STN), scaling, low-level feature extractor, high-level feature extractor, fully connected, and the final concatenation block. STN is a simulator of an affine transformation in a 2D image. The STN is used for face alignment. The scaling block is the main block, the fundamental idea of our approach. The details about this block are explained in the next subsection. After the scaling block, there are several internal outputs, each of which is one image of the original input, but in different scales, and hence has different sizes. Low and high-level feature extractors are two usual parts in most of the CNN. The fully connected block includes several fully connected layers and dropout layers. Finally, we combine all branch outputs with a late fusion technique.

A. THE SPATIAL TRANSFORMER NETWORK (STN) BLOCK

The STN was introduced by Jaderberg *et al.* [29] and Dai *et al.* [30]. The main idea of STN is to align the input by learning the transformer. This block is comprised of three parts: localization net, grid generator, and a sampler [29]. The localization net has several convolution layers, and finally, a fully connected layer to output θ, where θ is a matrix size

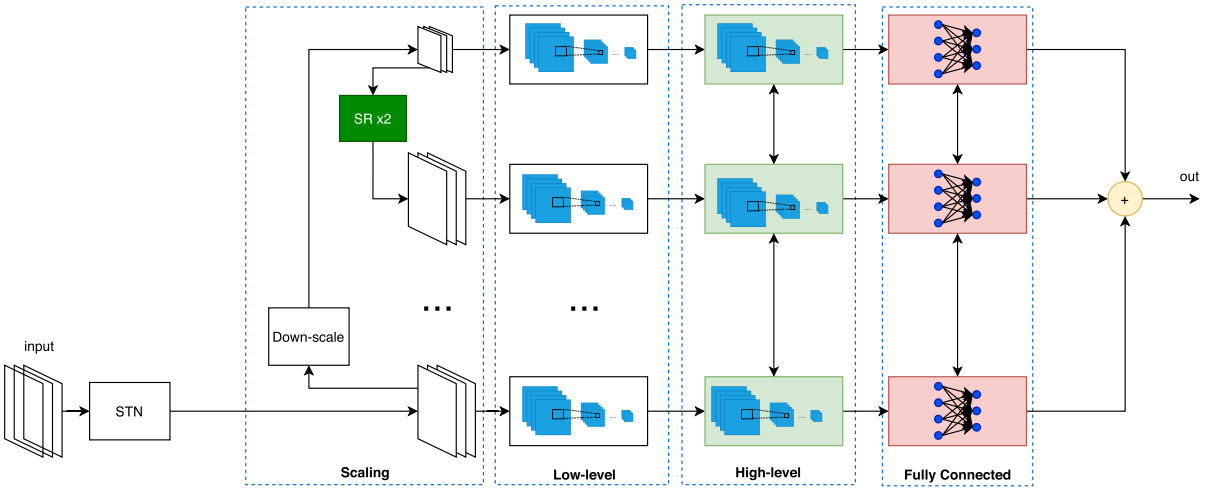


FIGURE 3. Overall network architecture.

of 2 × 3, a representation of an affine transform in a 2D image. The grid generator then accepts θ and makes a grid, and finally, the sampler uses this grid and generates the output image. The output image is from the input image with rotate, scale, and transforms operators. The input and output of this block are images with the same size and the same number of channels.

Different from in-the-lab images, ITW images are very different from the head pose direction. We add the STN block to help the network learn to align the face and make it easier to recognize.

Our implementation details follow the previously published paper [29]. Table 1 shows the details of the internal layers of this block. For the convolution layers, the parameters are the input channel, output channel, kernel size, and stride. The kernel size and stride are needed for the maxpool2d layer. For the linear layer, only two parameters are needed: the number of input nodes and that of output nodes. After the localization, the feature map is flattened and passed through the fully connected part. Our algorithm calculates the size of the feature map dynamically based on the input size. So, the block is adaptive to different sizes of the input images.

TABLE 1. The details of STN block.

Part	layer/function	params
localization	conv2d	(n_channel, 8, 7, 1)
	maxpool2d	(2, 2)
	relu	
	conv2d	(8, 10, 5, 1)
	maxpool2d	(2, 2)
	relu	
fc_loc	Linear	(out_size, 32)
	relu	
	Linear	(32, n_label)

B. THE SCALING BLOCK

The scaling block is the leading block in our architecture. The main idea of this block is to view the input image on different scale from small to large. Belong to that, super-resolution was used to upscale the image size. As in many CNNs, to ensure the efficiency of memory and computing, the input images are kept at the same size. And to use the best information from the input images, they are passed to the network at the largest attainable size. The input size may be limited by the computational limit and based on each dataset. While passing the same size images, as in the first observation, many of them were in low-resolution and were up-scaled by using some traditional algorithm. However, our approach down-scales them and then up-scales them again using the SR technique. This block is to view the overall context in the low-resolution images, along with the high-resolution image to consider the original features.

In the scaling block the network branches to three or more sub-networks. All sub-networks work with the same input image but on a different scale. The latest branch received the original input images, which had the highest resolution for the network. Due to the computational limit, most studies in the field of image classification use the input image from 100 to maximum 312. For the larger input size, the higher resolution does not improve the performance. For the batch mode, all images were resized to the central size before being passed through to the network. The larger image size is then down-scaled, and the smaller images needed to be up-scaled. We call the original input size $W \times H$. This process of scale input is the traditional algorithm, such as Nearest-neighbor interpolation, bilinear, and bicubic. While the down-scaled image is safe, the up-scaling from small size images to the larger size using the traditional algorithm is complicated and inaccurate. Our approach tends to overcome this issue. The first branch is applied to the lowest resolution image, which was down-scaled from the original input by the simple operator using mean pooling to implement. We declare the value *step* and *kstep* for the step scale value between two neighbors. By the limit of DL, *step* is set to 2. A large *kstep* can be used, but due to the computational limitation, we restrict *kstep* to only 1 or 2. The size of image for the first branch is

$$\frac{W}{2^{kstep}} \times \frac{H}{2^{kstep}}$$

Between the first and the last branch, there are *kstep* of SR branches, each of which is a SR block with the scale size of 2, 4, 8, ... from the lowest resolution image from the first branch. The size of i^{th} SR is given by equation 1.

$$\frac{W}{2^{kstep-i}} \times \frac{H}{2^{kstep-i}} \tag{1}$$

In case $k = 1$, there is only one SR branch in the scaling block, and the output size is the same as the original input size. In case $k = 2$, there are two SR branches, which have the sizes of $[W/2, H/2]$ and $[W, H]$. Our setup always ensures that the last SR part has the same size as the original input size. For the SR task, we use the EDSR architecture introduced by Lim *et al.* [27].

By learning how to resample the image size, we assume that this block can add useful information to this particular task, and thereby increases the accuracy of the prediction model.

C. LOW AND HIGH-LEVEL FEATURE EXTRACTOR

Typically, low and high-level feature extractors are combined in a base network architecture. We choose VGG16 [22] as the base network because this network is still used as the base of many recent network for the FER task [31]–[33]. From the base network, VGG16 [22], we separated into two parts for two levels of input. The low-level feature extractor receives the images as input and generates the feature map corresponding to the data. This block works at low level of features, e.g., edge, corner, and so on. The high-level feature extractor receives the feature map from the low-level part and makes a more in-depth, high-level features for the input.

While the input is passed through both extractors in this order, we separated them as two to share across branches. As in the second observation, we know that the CNNs are very sensitized with the input size, and here, each branch has different input sizes. The low-level features for each branch are quite different and cannot be shared because sharing low-level layers damages the network. The high-level feature block is in another environment. At this level, a high-level feature needs to be learned and is less dependent on the size of the input. Then the weight of this block can be shared across branches. The shared weights also act in a similar way to multi-task learning where the combination helps each task obtain better results.

The position of the cutting point denotes *pos*, which is the position of the convolution layers in the base network, where we separate the two parts. A lower *pos* value means that all branches share the weights in most of the internal layers, while the highest value of *pos* separates all branches. From the second observation, we assume that the low *pos* value degrades the network. Since the base network is VGG16, which has 12 convolution layers, the cutting position *pos* should be in $0 - 12$, which is the position of corresponding convolution layers. We analyze the effect of the cutting point (the *pos* value) in the experiments.

D. FULLY CONNECTED BLOCK AND CONCATENATION BLOCK

The fully connected block includes two fully connected layers (Linear, FC) and several additional layers. The output feature from the high-level block then passes through this block to get the vector to represent the score for each label. Depending on the experiment, we use either seven or eight emotions, and then the output vector sizes are set to seven or eight, respectively. We also use BatchNorm1d for the last feature map, and two dropout layers with p values of 0.25 and 0.5 for the first and the second FC layers, respectively. The ReLU activation function was applied after the first FC layer.

Similar to the high-level feature extractor block, the fully connected block was also shared among branches.

All branches were fused with the weighted late fusion strategy. The weight of each branch has been determined according to the contribution to the final score of the whole network.

III. THE PRIOR DISTRIBUTION LABEL SMOOTHING (PDLS) LOSS FUNCTION

FER for basic emotion is a classification problem, where each input image is classified into one of seven or eight classes. Softmax Cross-Entropy is the most popular loss function for classification tasks. The cross entropy (CE) loss function is given in equation 2.

$$CE = -\sum_{c \in C} t_c * log(\sigma(z_c)) \tag{2}$$

where:

- CE: cross entropy
- C: set of classes (labels)
- t_c: the distribution value of the label c in the ground truth where $\sum_{c \in C} t_c = 1$
- $\sigma(z_c)$: softmax function for z_c
- z_c: raw value score for class c from the model

In the real world, it is difficult to get the ground truth distribution for the labels for each sample; therefore, the *all in one* assumption was used in most cases. In the ideal case, the sample belongs to *one and only one* class; therefore, the one-hot vector is widely used in the classification task for labeling, so that equation 2 becomes the simple case of $-log(\sigma(z_k))$, where $t_c = 0$ for all $c \in C$ but the correct label k ($t_k = 1$). Then, all parts except the label k are omitted.

The Label Smoothing (LS) loss function has been introduced in other studies [34], [35], and [36]. The formula for LS is given as equation 3. The main idea here is the contribution of all incorrect labels. The parameter α was set around 0.9, meaning that the contribution for other labels is very small; e.g., for FER task, $|C| = 8$, then the weight for each of them is $0.1/8 \approx 0.0125$ and for the correct label is 0.9125. Although the weight of the incorrect labels is small, the LS has been used successfully in many classification tasks. The advantage of the LS over CE with one-hot is that all label scores predicted by the model are activated. Then the backpropagation

TABLE 2. The prior distribution of the emotions on the FER task.

	neutral	happiness	surprise	sadness	anger	disgust	fear	contempt
neutral	**0.777**	0.033	0.017	0.114	0.026	0.007	0.004	0.021
happiness	0.043	**0.918**	0.023	0.005	0.004	0.002	0.002	0.004
surprise	0.058	0.036	**0.787**	0.010	0.014	0.003	0.089	0.002
sadness	0.170	0.013	0.009	**0.748**	0.023	0.011	0.016	0.010
anger	0.075	0.025	0.060	0.032	**0.741**	0.035	0.023	0.009
disgust	0.090	0.017	0.017	0.089	0.125	**0.621**	0.008	0.034
fear	0.043	0.012	0.169	0.073	0.027	0.013	**0.659**	0.004
contempt	0.148	0.022	0.010	0.043	0.034	0.060	0.006	**0.678**

process can learn not only how to increase the score for the correct label but also how to decrease for the incorrect ones.

$$LS = -log(\sigma(z_k)) * \alpha + -\sum_{c \in C} log(\sigma(z_c)) * \frac{(1-\alpha)}{|C|} \quad (3)$$

where:

- $|C|$: size of label set
- α: parameter control the weight for each part

In the LS loss function, all labels except the correct one are given equal, i.e., they have a small role and are all equal. LS can be used extensively in many tasks when there is no information about the distribution. However, in many tasks like FER, for a particular correct label, the confusion to other classes are not uniform. The FER task has two advantages: the number of labels is small, just seven or eight and, more importantly, we know that for the particular label, the confusion for some specific classes is higher than others. For example, the correct label *fear* is very likely to be confused with *surprise* than with *disgust*. Another example is the *disgust* facial, which can easily be mistaken as *neutral*, or *sadness* than *anger* or *fear*. If we have this prior knowledge, the smoothing part should not be a uniform distribution. So, we proposed an extended version of LS with additional prior knowledge of the label's confusion call PDLS. The PDLS loss function was given by two parts: the one-hot and the prior distribution, as shown in equation 4.

$$PDLS = -\sum_{c \in C} (t_c * \alpha + d_{kc} * (1-\alpha)) * log(\sigma(z_c)) \quad (4)$$

where:

- α is a parameter to control the weight of one-hot and distribution.
- d_{kc} the prior distribution for the correct label k and the confusion label c.

All notations in equation 4 are similar to those in equation 2 and 3. The d_{kc} value is the new operand in this formula, and it replaced the uniform distribution $\frac{1}{|C|}$ in the LS loss function. The d matrix has the following properties:

$$size = |C| \times |C|$$
$$\sum_{c \in C} d_{kc} = 1, \quad \forall k \in C$$
$$argmax(d_{k1}, d_{k2}, \ldots, d_{k|C|}) = k, \quad \forall k \in C$$

The most important part is how to calculate the d_{kc}. Using Barsoum *et al.* [19], when correcting the labels for the FER2013 dataset [20], the authors of FER+ also provided the labels distribution information for every sample. In FER+, each sample was labeled by ten people, who need to classify each image to eight basic classes plus two additional classes, *unknown* and *non-face*. While the correct label's distribution for each sample is difficult to obtain, we assumed that the method for making the FER+ is a good approximation for the ground truth distribution. For each sample $s \in S$, S is the FER+ dataset, we have the approximate distribution ad_s. Since unknown and non-face images are omitted, we only use information for eight basic emotions, denote by E. Then ad_s is a vector in R^8 when 8 is the size $|E|$, and $\sum ad_s = 1$. Equation 5 is to calculate the average distribution for each ground truth emotion k. In this calculation, we used only the training set in the FER+.

$$d_k = \frac{\sum_{s \in S_k} ad_s}{|S_k|} \quad (5)$$

where:

- d_k: the average distribution of the label k, $d_k \in R^8$
- $|S_k|$: the size of the subset S_k, $S_k \subset S$, where the ground truth emotion is k. $\cup_{k \in E} S_k = S$.

The final prior distribution d_{ki} for the FER task is provided in table 2. Each row in the table is d_k, and k is one in eight emotion labels. The columns are the confusion labels, and there are also eight emotion labels. E.g. $d_{neutral,sadness} = 0.114$ means when image in *neutral*, there is 11.4% chance to confuse it as *sadness*. The number in the main diagonal is always higher than 0.5 that represents the distribution for its own emotion. The *happiness* emotion is very clear and easy to detect: $d_{happiness,happiness} = 0.918$, whereas *fear* and the *disgust* are difficult to detect and easy to be confused.

IV. DATASETS

There are three popular ITW datasets for the FER task, including the FER+ [19], RAF-DB [11], [12] and AffectNet [13] datasets. In this study, the experiments are conducted with all of them. The eight discrete emotions for the classification are *neutral*, *happiness*, *surprise*, *sadness*, *anger*, *disgust*, *fear* and *contempt*. Some previous datasets and studies used seven of them as they excluded *contempt* because it is difficult and rare in the real world. The details for each dataset are given below.

FER+ dataset. The FER+ dataset [19] is the first ITW dataset among them. The original version is the

TABLE 3. Number of images in training/testing/validation subsets of the FER+, RAF-DB, and AffectNet datasets.

	FER+			RAF-DB		AffectNet	
emotion	**train**	**test**	**valid**	**train**	**test**	**train**	**valid**
anger	2,054	271	284	705	162	24,882	500
disgust	107	15	23	717	160	3,803	500
fear	510	78	60	281	74	6,378	500
happiness	7,335	895	872	4,772	1,185	134,415	500
neutral	9,030	1,102	1,207	2,524	680	74,874	500
sadness	3,047	384	348	1,982	487	25,459	500
surprise	3,173	402	417	1,290	329	14,090	500
contempt	115	13	14	-	-	3,750	500
Total	**25,371**	**3,160**	**3,225**	**12,271**	**3,068**	**287,651**	**4,000**

FER2013 [20] by Goodfellow *et al.*, released for the ICML 2013 Workshop on Challenges in Representation Learning. But as the labeling accuracy of the FER2013 dataset is not reliable, Barsoum *et al.* reassigned the labels [19]. Ten people assigned manually the basic emotion for each image in the FER2013 dataset. The subset of the original images was excluded if it is classified as *unknown* or *non-face*. The final emotion label was assigned based on the voting from the ten people. The number of people voting for each emotion for each image was given, which was then used to calculate the approximate distribution of the emotion over that image.

The dataset includes all the images, each of which has one person's face aligned. The dataset images were collected from the Internet by querying many related expression keywords. There are many kinds of face in the real-world environment, and their pose and rotation make them more challenging to recognize. The images were aligned and centered, and they were scaled slightly differently. All images are low-resolution and in grayscale with a size of 48 × 48 pixels. Each corresponding label for each image is also given. The eight basic emotions are used in this dataset.

Table 3 and figure 4 show the distribution of train, test and validation on the FER+ dataset. The number of *neutral* images is highest, 9,030 on the train set, and 1,102 on the test set. The *disgust* emotion has the lowest number of images: only 107 on train and 15 on test. The *contempt* emotion has a similar number of images with *disgust*: only 115 on train and 13 on test. *Disgust*, *contempt* and *fear* have few images, compared with the other five emotions. This is normal in natural communication where people are usually in *neutral* and *happy* state and only rarely experience *disgust*, *contempt* or *fear*. Figure 4 shows that the distribution of emotions on training, testing, and validation on the FER+ are similar.

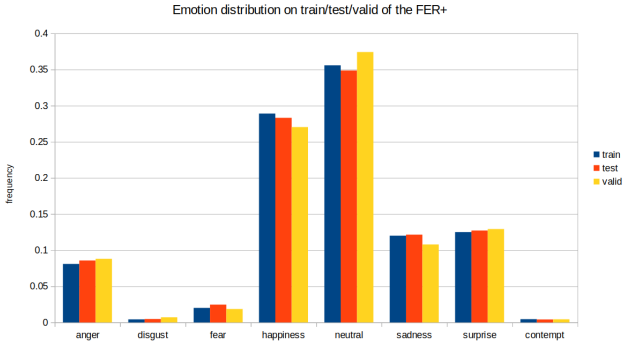


FIGURE 4. The FER+ data distribution of train/test/valid.

RAF-DB dataset. Shan Li, Weihong Deng, and Jun-Ping Du provided the Real-world Affective Faces Database (RAF-DB) for emotion recognition [11], [12]. The dataset contains about 30,000 images downloaded from the Internet. About 40 trained annotators labeled carefully the image. The dataset has two parts: the single-label subset (basic emotions) and the two-tab subset (compound emotions). We used the single-label subset with seven classes of basic emotions. This subset has 12,271 samples in the training set and 3,068 in the test set. The number of samples for each emotion is given in table 3. Notably, the RAF-DB dataset does not include the *contempt* expression. Figure 1 shows that images sizes in the RAF-DB vary from tiny to large, which makes it difficult for the DL model to deal with.

AffectNet dataset. The AffectNet [13] is the largest dataset for the FER task. The dataset contains more than one million images queried from the Internet by using related expression keywords. There are about 450,000 images manually annotated by trained persons. It also includes train, validation, and test sets. The test set has not yet been published, so most previous studies used validation set as the test set [13], [37]–[40]. Because the *contempt* emotion is rare in the natural world, some studies [40] used only seven emotions while other studies [13], [38], [39] analyzed all eight emotions. Another study used both eight and seven expressions [37]. Therefore, to compare our results with the previous studies, we performed experiments with both eight classes and seven classes.

Table 3 shows the number of samples for each emotion class on each subset train, validation, and test on the FER+, RAF-DB, and AffectNet datasets. The name they use for labels are a little different but can be mapped to the eight basic emotions as the emotion column. The FER+ has three separate subsets for training, validation, and testing, while two others have only two subsets. The AffectNet dataset has not published the testing subset, so as for most of the studies in this dataset, the validation is taken as the testing subset, and the validation subset during the training process should be randomly selected from the training subset. Similar to the RAF-DB, the training subset is randomly separated and then applied to get the training and validation subsets. Only AffectNet exhibits balanced validation (as the testing),

while the FER+ and RAF-DB are highly unbalanced. Both the FER+ and AffectNet datasets have eight emotions labels, and the RAF-DB has only seven emotion classes without *contempt* emotion expression.

Figure 5 gives some sample images for each class from the three datasets. In this figure, each column presents one emotion expression. The images in the first two rows (figure 5a) is from the FER+ dataset, figure 5b is from RAF-DB, and the rest (figure 5c) are from AffectNet. The last column of RAF-DB is empty because the RAF-DB dataset has seven emotions without contempt expression.

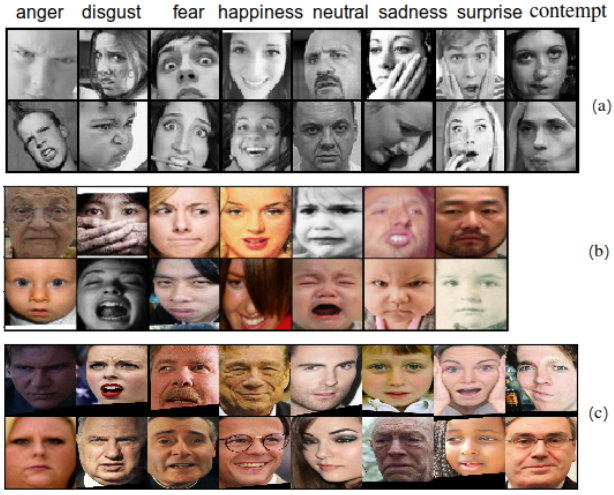


FIGURE 5. Sample images from the (a) FER+, (b) RAF-DB and (c) AffectNet datasets.

V. EXPERIMENTS AND RESULTS

This section reports our experiments and results. Subsection V-A gives the experimental setup. Results are shown in subsection V-B. Finally, subsection V-C presents the discussion about our approach and limitations.

A. EXPERIMENTAL SETUP

For all experiments, Fastai [41] and PyTorch [42] were used. Those toolboxes make DL experiments easier, with many build-in classes, functions, and also pre-trained models to reuse.

In DL, the network initialization has a significant impact on the training process. Commonly, weights are initially random. Having a good initialization strategy helps the networks to learn better and more smoothly. In our case, we carefully initialize the network weights. The STN block was set to identical transformation. The SR layers were initialized from previously published pre-trained model [27]. The base network, VGG16, was trained with different scale input images. The model weights were then saved and reloaded to our architecture. The careful initialization step has several advantages. It is easier to train the network, gives quicker network coverage, and makes a more stable network, leading to fewer variants.

We use Adam optimization algorithm [43] with an adaptive learning rate using The One Cycle Policy suggested by Smith [44]. The learning rates were set to 1e-3 for some later layers of the network, and 1e-4 for the STN block. The lower learning rate for STN with the transformation aims to keep this bock with little change.

The validation set is used to optimize the hyper-parameters, and then we collected the best models. The hyper-parameters for all our experiments include the learning rate and the number of epoch where the network gets the best result. Those models were used to evaluate the test set. We applied Test Time Augmentation on the test step. Eight randomly rotated, zoomed images are generated from each image and then passed through the model to get the raw score to predict. The final raw score is the average of their outputs.

For basic emotions recognition, several metrics are used to evaluate the results. The first and most widely used metric is *accuracy*, or *weighted accuracy* (WA), which is the number of correct answers divided by the total number of the test samples. But, when the number of samples for each class is highly unbalanced, WA may have poor performance, particularly FER task, because the emotions in the real world are usually unbalanced. Some emotions such as neutral, happy, or sad are more common than disgust, fear, or contempt. In this case, *unweighted accuracy* (UA) should be considered for the additional evaluation of the system. The UA metric is an unbiased version of WA. The UA is calculated by the average of the accuracy of each class. For comparison with other studies, both WA and UA are adopted in the experiments.

All experiments were run on Ubuntu 18.04, 32G RAM, GeForce RTX 2080 Ti GPU with 11G GPU RAM.

B. EXPERIMENTAL RESULTS

We report the experimental results for the RAF-DB, FER+, and AffectNet datasets.

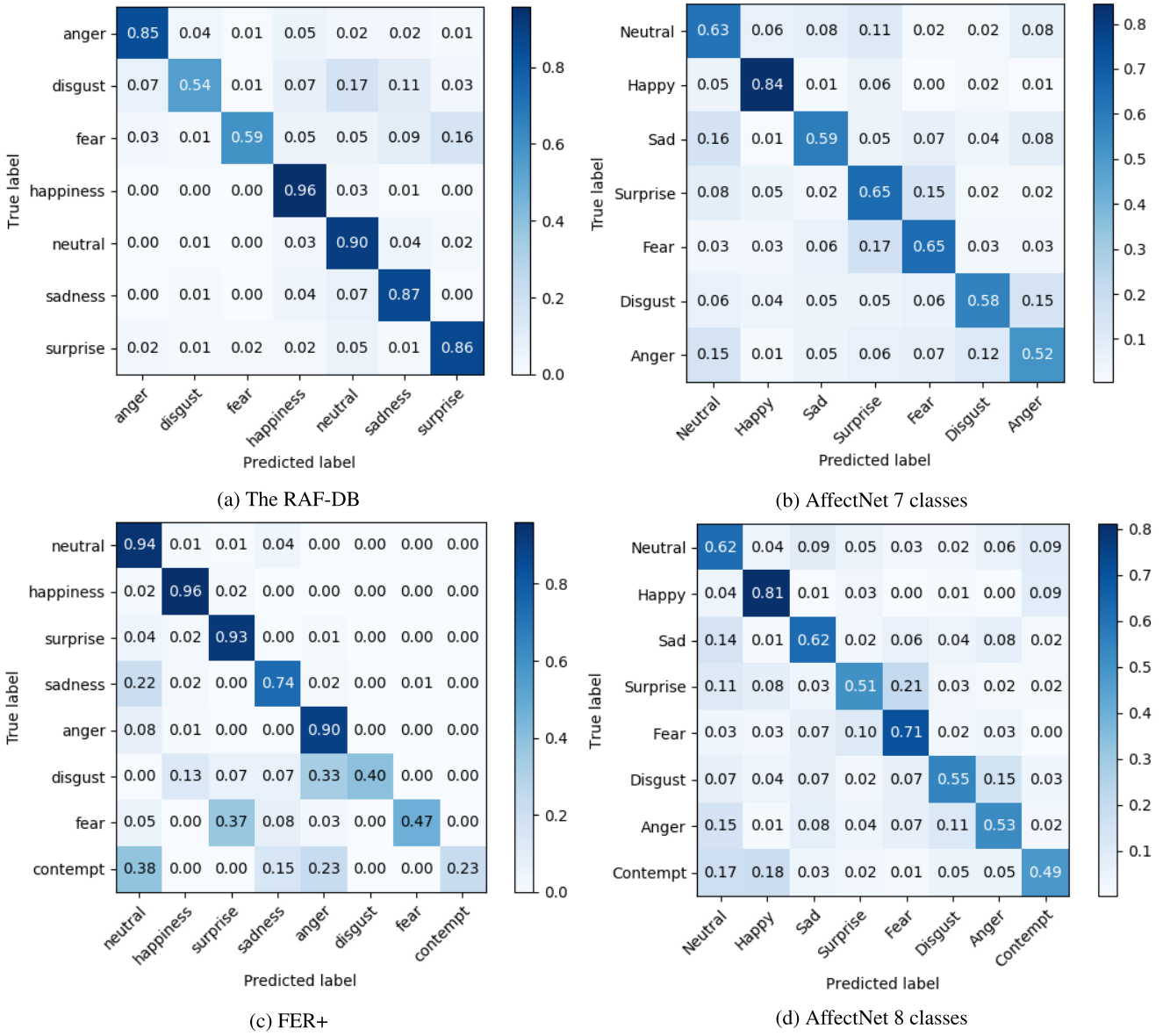
1) RAF-DB DATASET

Table 4 gives the results for the RAF-DB dataset. In previous studies, the methods in [38], [39], [45] report results in WA metric, and others [46], [47] report UA metric. We report and compare with previous findings in both WA and UA metrics. Our approach produces significantly better results than the recent studies on both metrics. For WA, we get 88.98%, which is improved by more than 2% in absolute terms or 2.4% relatively, compared to Wang *et al.* [39]. In the UA metric, our approach is 4.05% better in absolute terms compared to [46] or 5.28% relatively.

Figure 6 shows the confusion matrix for the RAF-DB. It is shown that the model gives very good accuracy for *happiness* and *neutral*, but the results for *disgust* and *fear* are only 54% and 59%, respectively. *Disgust* images were predicted as *neutral* by 17% and *fear* was predicted as *surprise* by 16%.

2) FER+ DATASET

Table 5 shows the experimental results on the FER+ test set. The highest accuracy is from the PSR model, which achieved



(a) The RAF-DB

(b) AffectNet 7 classes

(c) FER+

(d) AffectNet 8 classes

FIGURE 6. The confusion matrix on the test set for the RAF-DB, FER+ and AffectNet datasets.

TABLE 4. RAF-DB accuracy comparison (%).

Method	Year	WA↓	UA↓
PSR (Our)	2020	**88.98**	**80.78**
Region Attention [39]	2020	86.9	-
IPA2LT [38]	2018	86.77	-
Multi-region Ensemble [46]	2018	-	76.73
Base-line (VGG16)	-	85.89	76.27
PAT-CNN [45]	2018	84.2	-
CFL [47]	2018	-	75.73

TABLE 5. FER+ accuracy comparison (%).

Method	Accuracy↓
PSR (Our)	**89.75** (89.56 ± 0.03)
SENet Teacher [48]	89.10 (88.80 ± 0.30)
CNNs, BOVW + SVM [37]	87.76
ResNet late fusion (SVM) [16]	87.40
Base-line (VGG16)	85.89
Probabilistic label drawing [19]	85.35

89.75%. Compared to the best previous result in the literature by Albanie *et al.* [48], our approach is improved by 0.65%.

The average accuracy for our proposed architecture is **69.54%** and F1 score (macro) is **74.88%**. The low accuracy on *disgust* and *fear* makes the F1 score and average accuracy far lower than the average. Future work should consider focusing on increasing the number of sample of *disgust* and *fear* to improve the accuracy for these two expressions.

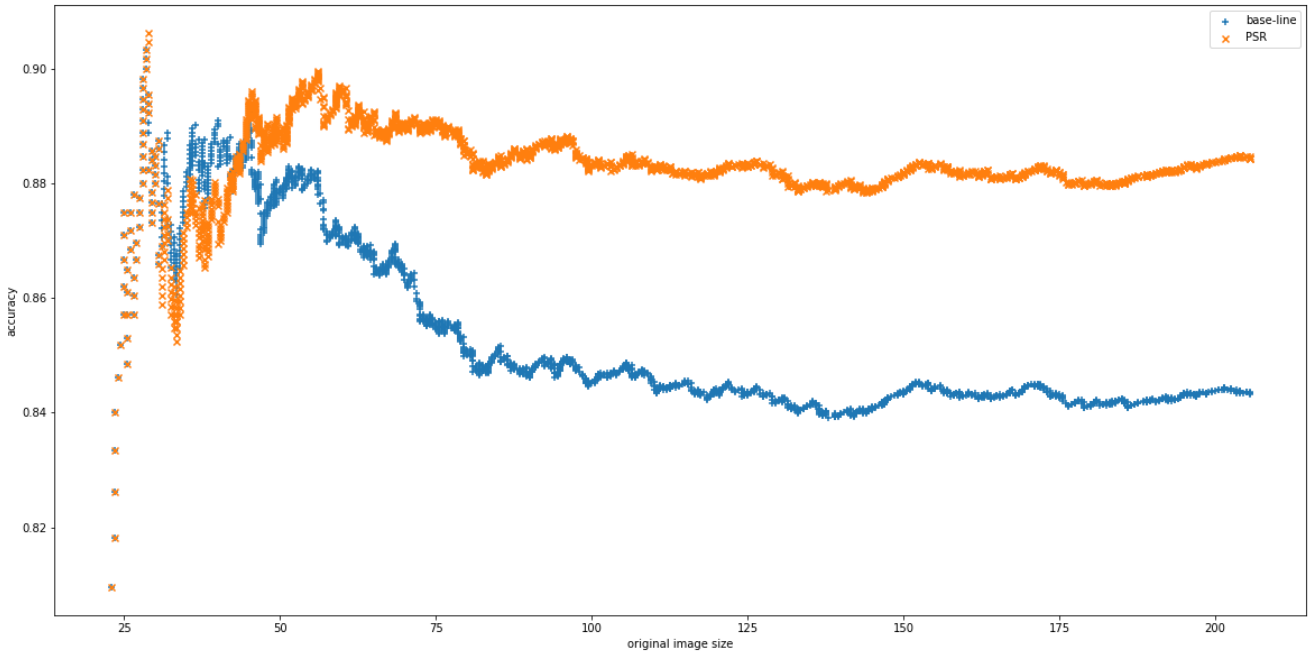


FIGURE 7. Cumulative accuracy by size on the test set of RAF-DB dataset with the VGG16 (base-line) and the PSR architecture.

Figure 6c shows the confusions matrix on the test set of the PSR architecture: *happiness* has the highest accuracy of 96%, followed by *neutral*, *surprise* and *anger*. All four expressions had accuracy above 90%. The lowest accuracy was for *contempt*, 23% accuracy. Due to the lack of contempt images, the model could not learn to distinguish it from *neutral*, *anger*, or *sadness*. Some emotions have high likelihood of wrong classification: *fear* predicted as *surprise* by 37%, *disgust* classified as *anger* by 33% and *sadness* classified as *neutral* by 22%. These high levels of confusion are typical in the real world because even for humans, it can be difficult to distinguish these pairs of emotions.

3) AffectNet DATASET

We compared both eight and seven classes in the AffectNet dataset. Table 6 shows the results in classification accuracy (WA). In the classification of eight emotions, our model archived the accuracy of 60.68%, outperformed the current state-of-the-art 59.58% achieved by Georgescu *et al.* [37]. In the seven-emotion task, our model archived the accuracy of 63.77%, slightly improved relative to the current highest one at 63.31% [37]. Figure 6b and figure 6d present the confusion matrix for the AffectNet in the seven-class task and eight-class task, respectively. The *happy* expression has the highest detection rate in both cases, followed by the *fear* emotion. *Surprise, anger, and disgust* have a similar performance in both cases. In the eight-expression task, *contempt* has the lowest performance just at 49%.

Figure 7 shows the cumulative accuracy according to the size of the original image on the base-line network and the PSR architecture. The PSR was run with the three branches

TABLE 6. AffectNet accuracy comparison (%).

Method	Year	8 classes↓	7 classes↓
PSR (Our)	2020	**60.68**	**63.77**
LLHF [37]	2019	59.58	**63.31**
AlexNet [13]	2020	58.00	-
IPA2LT [38]	2018	57.31	-
Region Attention [39]	2020	52.97	-
HERO [40]	2019	-	62.11

[1, 2, 1] and the cutting point at the sixth convolution layer, with the original input size of 100 pixels. The image size ranged from 23 pixels to about 1200 pixels. Because the large images were resized to a fixed size at 100 pixels, we consider only those images smaller than 100 pixels to see how our approach is affected. We omitted the first twenty points because they are unstable to calculate the accuracy. The figure shows that initially, with tiny image sizes less than 40 pixels, both base-line and PSR are unstable. But after 40 pixels, the PSR architecture is improved and works better than the base-line network. The PSR maintained this trend to the end of the dataset because, in our approach, we added the super-resolution module with double size for a small image in one of the three branches, and another branch for half size $100/2 = 50$ improved the recognition accuracy.

Figure 8 shows the accuracy discrepancy by size between PSR and VGG16 on the test set of the RAF-DB dataset. The blue points are raw values, and yellow ones are the smoothed version. The accuracy discrepancy represents the speed of the improvement of the PSR over the baseline network. It is clear that the improvement had the highest speed when the

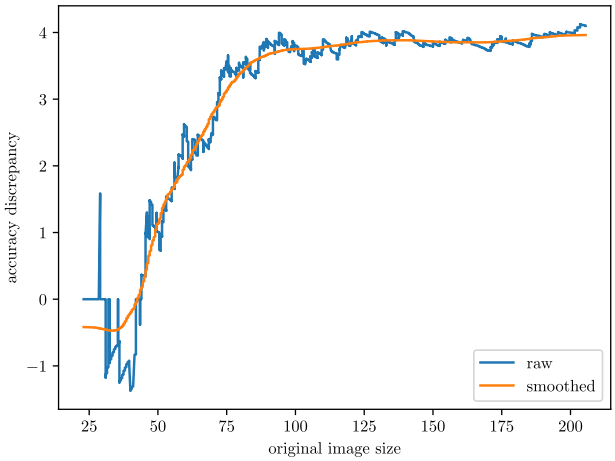


FIGURE 8. The discrepancy of accuracy by size on the test set of the RAF-DB dataset between PSR and baseline.

TABLE 7. Analysis of the effectiveness of each block on the RAF-DB (%).

Method	WA↓	UA↓
PSR full STN and pyramid	**88.98**	**80.78**
PSR without STN	88.62	79.57
Base-line (VGG16)	85.89	76.27

original image size ranged from 40 pixels to about 55 pixels; it slows down when the size reached between 55 pixels and 75 pixels, and it becomes lower for 75-85 pixels. After 85 pixels, the improvement continues but at a slow speed. Notably, in the experiments for RAF-DB, the original input size is 100 pixels resolution, then 50 pixels is half of the input size.

4) THE EFFECTIVENESS OF EACH BLOCK

Table 7 shows a comparison between some variations of the PSR on the RAF-DB dataset. The second row presents the result of the PSR without the STN block, which means that there are only the pyramid structure on top of the baseline network with three branches ($kstep = 2$). It is clear that on both WA and UA metrics, this network architecture gets better results than the VGG16. The improvements are significant in both cases of metrics, 2.73% for WA and 3.30% for UA. This implies that our pyramid with SR has an important role. When adding STN block to make the full PSR architecture, we can get a little improvement, about 0.36% in WA and 0.81% in WA metrics. We analyzed the effectiveness of the super-resolution reconstruction module by breaking down the PSR without the STN block to three separate branches to see the contribution of each to the final fusion. Figure 9 shows the accuracy of each separated branch and also the fusion of them on the PSR architecture. As expected, the small size branch got the lowest accuracy, and the fusion gets the best one when combining all three branches. The SR branch and original input size use the same scale input size, one is SR from the haft size and another is the original input size. Although using the same scale size, the SR branch performs better

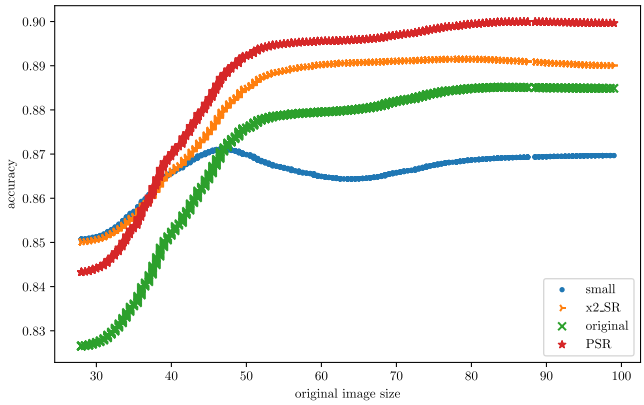


FIGURE 9. The accuracy by original image size on each branch of the PSR without the STN block on the RAF-DB test-set.

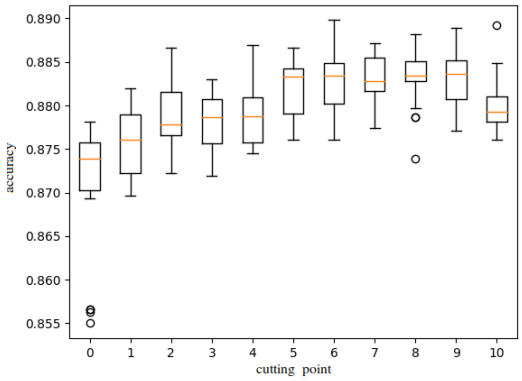


FIGURE 10. The boxplots of performance with different cutting points (accuracy).

than the original size branch. The discrepancy between SR and original input size branch is large for the small images, and it decreases as the size increase. The results clearly reconfirm that the SR branch helps the network improve the performance when the original image size is small.

Figure 10 shows the performance on the RAF-DB dataset by the cutting point *pos* of the convolution layers from VGG16. The network exhibits the lowest performance at the point $pos = 0$, indicating that all the convolution layers are shared. The accuracy increases as the *pos* value increases but this improvement ceases after the particular cutting point at $pos = 5$. After the fifth layer cutting, the accuracy remains stable around the particular value. This result supports the second observation, i.e. the CNN is sensitized with the input size. Sharing some early convolution layers causes the network to crash. On the other hand, the deeper layers can be shared because the former convolution layers are learning the low-level features, and the later convolution layers are working on more abstract, high-level features.

5) THE SENSITIVITY OF THE NETWORK TO THE DIFFERENT INPUT IMAGE SIZE

Figure 11 shows the comparison between PSR and VGG16 about the sensitivity when changing the input image

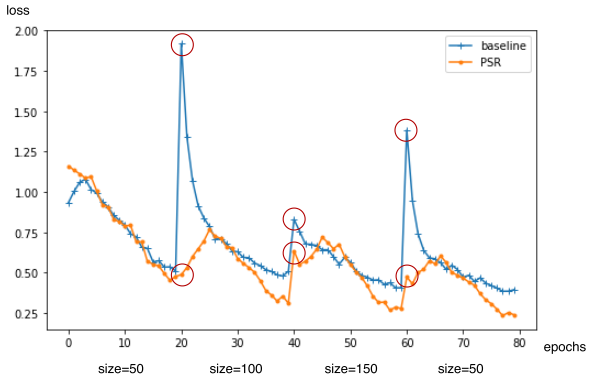


FIGURE 11. Visualization of training loss of the PSR and baseline during the training process of the RAF-DB when changing the original input size in the sequence 50, 100, 150, and back to 50 pixels again.

TABLE 8. The loss function comparison (accuracy %).

		CE	LS	PDLS
RAF-DB	VGG16	85.66	**85.89**	85.77
	PSR	87.78	88.56	**88.98**

size on the RAF-DB dataset. The training process is similar as in figure 2a with the first 20 freeze steps were omitted. The changing points are in epoch 20, 40, and 60. The graph exhibits that the PSR is less sensitive than the baseline. After the changing point, the loss of PSR architecture is slightly increasing. But the VGG16 has a large increase of loss values. The results confirm that our approach has the robustness for the ITW FER task where the original image size varies, although the CNNs usually sensitized to the input image.

6) THE COMPARISON OF THE THREE DIFFERENT LOSS FUNCTIONS

Table 8 compares three loss functions, including the CE, LS and PDLS on the RAF-DB dataset. For each type of loss function, we conducted on experiment on the baseline architecture, VGG16, and our proposed network architecture. In both cases of the VGG16 and PSR network architecture, the CE loss function gets the lowest accuracy. For the baseline network, the LS is slightly better than PDLS, by 0.12%. For the PSR architecture, however, the PDLS is slightly better than LS with a margin of 0.42%.

Figure 12 shows some sample images from the RAF-DB dataset that PSR predict the correct emotions while the baseline network gave the incorrect emotions. All three images are in the low-resolution which the size ranged between 45 and 56 pixels.

C. DISCUSSION

The experiments have demonstrated the significant improvement of our approach in FER task on all the three datasets. Compared to the base network, VGG16, our pyramid architecture with additional SR block and late fusion greatly improves the performance. On the RAF-DB dataset, our accuracy is better by about 2% in WA metric and 4.05% in

FIGURE 12. Sample images in low-resolution in the RAF-DB dataset where PSR recognizes better than the baseline network.

UA metric, compared to the state-of-the-art results. The most substantial improvement in accuracy has been obtained on the RAF-DB dataset. On the AffectNet dataset, PSR improves the accuracy by 1.01% and 0.46%, compared to the best previous study, respectively. Although the given input is in a small size (48 × 48) as the FER+ dataset, our PSR model generated better results. Among the three datasets, the RAF-DB exhibited the most improvement because the RAF-DB has many image sizes from 23-100. The AffectNet dataset shows less improvement. For the FER+ dataset, the dataset includes the resized and cropped version of images; using the original version, if it were available, PSR would give better results. Notable, the different accuracy discrepancies versus the second best algorithm in tables 4, 5, and 6 might be due to each of these tables having a different set of algorithms.. Overall, the pyramid with SR has a significant improvement for the FER task on the ITW dataset. The SR branch helps the network performance on the low-resolution image and then combining with other branches makes the whole network better. The STN block also has some improvement.

As in the second observation, the DL networks are sensitized with the image input size, and the low-level block in each branch is very different. The result shown in figure 10 supports our assumption. When the *pos* value is decreased, indicated that more layers are shared, including some low-level convolution layers, the network is degraded. When the *pos* value is increased, indicating that the low-level features are less shared, the network exhibits better results. Due to the trade-off between the performance and the computing cost in real practice, the results in figure 10 are useful for selecting the cutting point.

The experimental results in table 8 reconfirmed that LS loss function is better than CE as in many previous studies [34]–[36]. Both the LS and PDLS have better performance compared to CE, and in the case of the PSR architecture, PDLS shows a significant boost. The PDLS loss function gave a slight improvement over the original LS function in the FER task, but it varies case by case, it depends on the network architecture. In the case of VGG16, the experiments show that the PDLS is nearly equal to LS, which suggests that future improvements should be needed. The results from PSR model suggests that either LS or PDLS is good for the loss function in the FER task, instead of CE.

Despite the significant improvements presented in our study, some limitations warrant further research. The first is

the step of the scale-up from the lowest resolution. The pyramid architecture has viewed the input on several scales, but a step is an integer number larger than one, and 2 is a starting value. But, the double scale is still a tremendous value. While the scale 1.2 is a good point for most of the augmentation techniques, and $\frac{1}{1.2}(\approx 0.83)$ in the reverse case, we suggest that the scale step should be $1.2^2 = 1.44$, or approximated as 1.5. For the traditional algorithm, a decimal scaling value is possible, but it cannot be used for the DL approaches. The second weakness is the baseline network architecture. Although several network architectures, more reliable than VGG16, such as ResNet [49] and SENet [50] have been reported, we chose the VGG16 as the base network. Although our approach is general, we can apply many kinds of CNN, and the re-implementation is needed for each base network. Our approach is not a simple module, so extra efforts must be taken to implement case by case. The innovation of another architecture is left for future work.

VI. CONCLUSION

In this study, we addressed the various different-image-size problem in the FER task for ITW datasets, where the original input image size varies. Although the CNNs could work on the image with a small rotate and scale, they are worthless when the scale is enormous. The main contribution of this study is the development of a pyramid network architecture with several branches, each of which works on one level of input scale. The proposed network is based on the VGG16 model, but it can be extended to another baseline network architecture. In the PSR architecture, the SR method is applied for up-scaling the low-resolution input. Experiments on three ITW FER datasets show that our proposed method outperforms all the current state-of-the-art methods.

REFERENCES

[1] A. Mehrabian, *Nonverbal Communication*. New Brunswick, NJ, USA: Aldine Transaction, 1972.
[2] P. Ekman, "Are there basic emotions?" *Psychol. Rev.*, vol. 99, no. 3, pp. 550–553, 1992.
[3] P. Ekman, "Basic emotions.," in *Handbook Cognition Emotion*. New York, NY, USA: Wiley, 1999, pp. 45–60.
[4] J. A. Russell, "A circumplex model of affect.," *J. Personality Social Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980.
[5] P. Ekman and W. Friesen, *Facial Action Coding System*, vol. 1. Mountain View, CA, USA: Consulting Psychologists Press, 1978.
[6] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image Vis. Comput.*, vol. 27, no. 6, pp. 803–816, May 2009.
[7] L. Ma and K. Khorasani, "Facial expression recognition using constructive feedforward neural networks," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 34, no. 3, pp. 1588–1595, Jun. 2004.
[8] J. J. Lien, T. Kanade, J. F. Cohn, and C.-C. Li, "Automated facial expression recognition based on FACS action units," in *Proc. 3rd IEEE Int. Conf. Autom. Face Gesture Recognit.*, 1998, pp. 390–395.
[9] T. Zhang, W. Zheng, Z. Cui, Y. Zong, J. Yan, and K. Yan, "A deep neural network-driven feature learning method for multi-view facial expression recognition," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2528–2536, Dec. 2016.
[10] P. S. Aleksic and A. K. Katsaggelos, "Automatic facial expression recognition using facial animation parameters and multistream HMMs," *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 1, pp. 3–11, Mar. 2006.
[11] S. Li and W. Deng, "Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 356–370, Jan. 2019.
[12] S. Li, W. Deng, and J. P. Du, "Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 2584–2593.
[13] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Trans. Affect. Comput.*, vol. 10, no. 1, pp. 18–31, Jan. 2019.
[14] P. Liu, S. Han, Z. Meng, and Y. Tong, "Facial expression recognition via a boosted deep belief network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1805–1812.
[15] K. Liu, M. Zhang, and Z. Pan, "Facial expression recognition with CNN ensemble," in *Proc. Int. Conf. Cyberworlds (CW)*, Sep. 2016, pp. 163–166.
[16] C. Huang, "Combining convolutional neural networks for emotion recognition," in *Proc. IEEE MIT Undergraduate Res. Technol. Conf. (URTC)*, Nov. 2017, pp. 1–4.
[17] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, and A. M. Dobaie, "Facial expression recognition via learning deep sparse autoencoders," *Neurocomputing*, vol. 273, pp. 643–649, Jan. 2018.
[18] D. C. Tozadore, C. M. Ranieri, G. V. Nardari, R. A. F. Romero, and V. C. Guizilini, "Effects of emotion grouping for recognition in human-robot interactions," in *Proc. 7th Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2018, pp. 438–443.
[19] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," in *Proc. 18th ACM Int. Conf. Multimodal Interact.*, 2016, pp. 279–283.
[20] I. J. Goodfellow, "Challenges in representation learning: A report on three machine learning contests," *Neural Netw.*, vol. 64, pp. 59–63, Apr. 2015.
[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[23] C. Dong, "Learning a deep convolutional network for image super-resolution," in *Computer Vision*, vol. 8692, D. Fleet, Ed. Cham, Switzerland: Springer, 2014, pp. 184–199.
[24] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.
[25] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1874–1883.
[26] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.
[27] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1132–1140.
[28] Y. Hu, X. Gao, J. Li, Y. Huang, and H. Wang, "Single image super-resolution via cascaded multi-scale cross network," 2018, *arXiv:1802.08808*. [Online]. Available: http://arxiv.org/abs/1802.08808
[29] M. Jaderberg, K. Simonyan, A. Zisserman, others, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[30] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 764–773.
[31] M. Hu, H. Wang, X. Wang, J. Yang, and R. Wang, "Video facial emotion recognition based on local enhanced motion history image and CNN-CTSLSTM networks," *J. Vis. Commun. Image Represent.*, vol. 59, pp. 176–185, Feb. 2019.
[32] S. Li, W. Zheng, Y. Zong, C. Lu, C. Tang, X. Jiang, J. Liu, and W. Xia, "Bimodality fusion for emotion recognition in the wild," in *Proc. Int. Conf. Multimodal Interact.*, Oct. 2019, pp. 589–594.
[33] A. Sepas-Moghaddam, A. Etemad, F. Pereira, and P. L. Correia, "Facial emotion recognition using light field images with deep attention-based bidirectional LSTM," in *Proc. ICASSP - IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3367–3371.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[35] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.
[36] R. Müller, S. Kornblith, and G. Hinton, "When does label smoothing help?" in *Proc. NIPS*, 2019, pp. 4694–4703.
[37] M.-I. Georgescu, R. T. Ionescu, and M. Popescu, "Local learning with deep and handcrafted features for facial expression recognition," *IEEE Access*, vol. 7, pp. 64827–64836, 2018.
[38] J. Zeng, S. Shan, and X. Chen, "Facial expression recognition with inconsistently annotated datasets," in *Proc. Eur. Conf. Comput. Vis.*, vol. 11217, 2018, pp. 227–243.
[39] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao, "Region attention networks for pose and occlusion robust facial expression recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 4057–4069, 2020.
[40] W. Hua, F. Dai, L. Huang, J. Xiong, and G. Gui, "HERO: Human emotions recognition for realizing intelligent Internet of Things," *IEEE Access*, vol. 7, pp. 24321–24332, 2019.
[41] J. Howard. (2018). *Fastai*. [Online]. Available: https://github.com/fastai/fastai
[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS*, 2017, pp. 1–4.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[44] L. N. Smith, "Cyclical learning rates for training neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 464–472.
[45] J. Cai, Z. Meng, A. S. Khan, Z. Li, J. O'Reilly, and Y. Tong, "Probabilistic attribute tree in convolutional neural networks for facial expression recognition," 2018, *arXiv:1812.07067*. [Online]. Available: http://arxiv.org/abs/1812.07067
[46] Y. Fan, J. C. Lam, and V. O. Li, "Multi-region ensemble convolutional neural network for facial expression recognition," in *Artificial Neural Networks and Machine Learning* (Lecture Notes in Computer Science), vol. 11139. Berlin, Germany: Springer, 2018, pp. 84–94.
[47] F. Lin, R. Hong, W. Zhou, and H. Li, "Facial expression recognition with data augmentation and compact feature learning," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 1957–1961.
[48] S. Albanie, A. Nagrani, A. Vedaldi, and A. Zisserman, "Emotion recognition in speech using cross-modal transfer in the wild," in *Proc. ACM Multimedia Conf. Multimedia Conf.*, 2018, pp. 292–301.
[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[50] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Aug. 2020.

THANH-HUNG VO received the B.Eng. degree from the Ho Chi Minh City University of Technology, in 2010, and the M.Eng. degree from Vietnam National University, Vietnam, in 2013, all in computer science. He is currently pursuing the Ph.D. degree with the Pattern Recognition Laboratory, School of Electronics and Computer Engineering, Chonnam National University, South Korea. Since 2011, he has been working as a Lecturer with the Ho Chi Minh City University of Technology. His research interests include natural language processing, speech, and computer vision applying machine learning, and deep learning techniques.

GUEE-SANG LEE (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in computer engineering from Seoul National University, South Korea, in 1980 and 1982, respectively, and the Ph.D. degree in computer science from Pennsylvania State University, in 1991. He is currently a Professor with the Department of Electronics and Computer Engineering, Chonnam National University, South Korea. His primary research interests include image processing, computer vision, and video technology

HYUNG-JEONG YANG (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Chonbuk National University, South Korea. She is currently a Professor with the Department of Electronics and Computer Engineering, Chonnam National University, Gwangju, South Korea. Her main research interests include multimedia data mining, medical data analysis, social network service data mining, and video data understanding.

SOO-HYUNG KIM (Member, IEEE) received the B.S. degree in computer engineering from Seoul National University, in 1986, and the M.S. and Ph.D. degrees in computer science from the Korea Advanced Institute of Science and Technology, in 1988 and 1993, respectively. Since 1997, he has been a Professor with the School of Electronics and Computer Engineering, Chonnam National University, South Korea. His research interests include pattern recognition, document image processing, medical image processing, and deep learning applications.