Edge-AI-Driven Framework with Efficient Mobile Network Design for Facial Expression Recognition

YIRUI WU and LILAI ZHANG, Hohai University
ZONGHUA GU, Umeå University
HU LU, Jiangsu University
SHAOHUA WAN, University of Electronic Science and Technology of China

Facial Expression Recognition (FER) in the wild poses significant challenges due to realistic occlusions, illumination, scale, and head pose variations of the facial images. In this article, we propose an Edge-AI-driven framework for FER. On the algorithms aspect, we propose two attention modules, Arbitrary-oriented Spatial Pooling (ASP) and Scalable Frequency Pooling (SFP), for effective feature extraction to improve classification accuracy. On the systems aspect, we propose an edge-cloud joint inference architecture for FER to achieve low-latency inference, consisting of a lightweight backbone network running on the edge device, and two optional attention modules partially offloaded to the cloud. Performance evaluation demonstrates that our approach achieves a good balance between classification accuracy and inference latency.

CCS Concepts: • **Computer systems organization → Real-time systems**; • **Computing methodologies → Machine learning algorithms;**

Additional Key Words and Phrases: Deep learning, Facial Expression Recognition, edge computing, cloud offloading

ACM Reference format:
Yirui Wu, Lilai Zhang, Zonghua Gu, Hu Lu, and Shaohua Wan. 2023. Edge-AI-Driven Framework with Efficient Mobile Network Design for Facial Expression Recognition. *ACM Trans. Embedd. Comput. Syst.* 22, 3, Article 57 (April 2023), 17 pages.
https://doi.org/10.1145/3587038

Y. Wu and L. Zhang contributed equally to this research.

This work was supported by National Natural Science Foundation of China under Grant No. 62172438, Key Project of Shenzhen City Special Fund for Fundamental Research under Grant No. 202208183000751, National Key R&D Program of China under Grant No. 2021YFB3900601, Fundamental Research Funds for the Central Universities under Grant No. B220202074, Fundamental Research Funds for the Central Universities, JLU, Joint Foundation of the Ministry of Education under Grant No. 8091B022123, and the Kempe Foundation, Sweden.

Authors' addresses: Y. Wu and L. Zhang, Hohai University, Fochengxi Road 8, Nanjing City, Jiangsu Province, 210093, China; emails: wuyirui@hhu.edu.cn, zhanglilai1999@gmail.com; Z. Gu (corresponding author), Umeå University, SE-901 87, Umeå, 90187, Sweden; email: zonghua.gu@umu.se; H. Lu, Jiangsu University, Xuefu Road 301, Zhenjiang City, Jiangsu Province, 212013, China; email: luhu@ujs.edu.cn; S. Wan (corresponding author), University of Electronic Science and Technology of China, Guanguang Road 1301-78, Shenzhen City, Guangdong Province, 518028, China; email: shaohua.wan@ieee.org.

1539-9087/2023/04-ART57 $15.00
https://doi.org/10.1145/3587038

1 INTRODUCTION

Facial Expression Recognition (**FER**) is an active and relevant research topic in the computer vision field. For *FER in the wild*, the dataset consists of images captured from albums, videos, online pictures, and so on, in contrast to laboratory-collected datasets in a standard laboratory environment. FER in the wild poses significant challenges to FER algorithms, as the data collection method may cause significant image quality degradations, such as realistic occlusions, illumination, scale, and head pose variations, which often make it necessary for multiple human labelers to vote to determine the ground-truth labels. In contrast, a dataset gathered in a standard laboratory environment consists of images shot in a studio with good conditions, where the human models are asked to make certain expressions, so both the labeling task and the classification task are easier than FER in the wild.

Deep Neural Networks (**DNNs**) trained with Deep Learning have achieved significant success in many application domains, including FER. After model training, DNNs are typically deployed as cloud services by commercial vendors, relying on large servers in the cloud for serving a large number of inference requests. Edge devices, e.g., mobile phones or smart cameras, send images to the cloud, which runs the DNN inference task and returns the results to the devices. Cloud access across a wide area network alleviates the issue of resource constraints of edge devices, but it may not provide the optimal Quality-of-Service due to network latency and disruptions. The network transmission time and energy consumption may be unacceptably high, esp. for uplink transmission of large high-resolution images over low-bandwidth and/or unreliable wireless networks. Edge computing is an increasingly popular paradigm that aims to execute many tasks locally on local edge devices instead of offloading them to the remote cloud. With the increasing computing power of today's edge devices such as mobile phones and smart cameras, DNN inference is increasingly being performed locally on the edge device instead of in the cloud. This helps to ensure the real-time performance of tasks in time-sensitive application scenarios (e.g., FER of vehicle drivers), and also provides better privacy protection for the users. However, it may be infeasible or too slow to deploy large models directly on edge devices with limited computing and memory resources, making it necessary to offload all or part of the DNN computation to the cloud in some application scenarios. Hybrid edge/cloud computing is also a popular approach, where the workload is split and distributed between edge devices and the cloud, depending on runtime conditions such as network bandwidth and battery power.

In this article, we propose an Edge-AI-driven Framework for FER, with contributions on two aspects:

- On the algorithms aspect: we propose two attention modules: the ***Arbitrary-oriented Spatial Pooling (ASP)*** module for spatial attention performs 1D pooling on the feature map in four directions (horizon, vertical, forward diagonal, and backward diagonal) to encode long-range dependencies in different directions to address the issue of head pose variations; The ***Scalable Frequency Pooling (SFP)*** module for channel attention applies different levels of **Discrete Wavelet Transform** (**DWT**) to encode multi-scale frequency information and focus on dominant frequency components representing salient image contents, to address the issues of realistic occlusions, illumination, and scale variations. The combination of the two modules (ASP and SFP) can effectively encode both long-range dependencies and multi-scale frequency information in the feature maps and helps to improve the classification accuracy for FER in the wild.
- On the systems aspect: we propose an edge-cloud joint inference architecture for FER to achieve low-latency inference, consisting of a lightweight backbone network running on the edge device, and two optional attention modules partially offloaded to the cloud.

Intermediate feature maps of the backbone network are transmitted to and from the cloud, thus reducing the amount of data transmission and network latency compared to transmitting image pixels to the cloud. The joint inference architecture helps to reduce DNN inference latency compared to device-only and cloud-only execution.

The rest of the article is organized as follows. We present related works in Section 2; the details of our proposed Edge-AI-driven framework in Section 3, including the overall framework, and the detailed design of the ASP and SFP modules; performance evaluation results in Section 4; conclusions in Section 5.

2 RELATED WORK

2.1 Facial Expression Recognition (FER)

A FER system typically consists of three steps, i.e., face detection, feature extraction, and expression recognition. Face detection methods like CRFace [31] and HLA-Face [34] can locate faces accurately in extreme conditions such as crowds, low lighting, and so on. We do not consider the face detection problem in this article.

Many techniques have been proposed for feature extraction to capture the geometry and appearance characteristics of facial expressions. Liu et al. [18] extract robust deep salient features from saliency-guided facial patches, which are further fed into a conditional **convolutional neural network enhanced random forest** (**CoNERF**) for FER classification. By involving the idea of progressive refinement, Xie et al. [37] propose a Deep Attentive Multi-path CNN model, which not only adaptively emphasizes the features that are highly relevant to the FER task, but also encodes intra-class variations to improve accuracy. Li et al. [15] propose Patch-Gated CNN, which first decomposes an intermediate feature map into several patches according to the positions of related facial landmarks to determine the regions of interest, then uses the Patch-Gated Unit to reweigh each patch by the unobstructed-ness or importance that is computed from the patch itself.

For FER in the wild, Li et al. [16] propose a patch-based attention network for occlusion-aware FER, where the patches are cropped from the areas of eyes, nose, mouth, and so on, and then the selected 24 patches are fed into an attention network to assign weights for further global FER. Wang et al. [32] propose a **Region Attention Network** (**RAN**) to adaptively capture the importance of facial regions to handle occlusions and head pose variations. Zhao et al. [40] propose EfficientFace for label distribution learning, with a local-feature extractor and a channel-spatial modulator for extracting local and global-salient facial features. Zhao et al. [39] propose a global Multi-scale and local Attention network (MA-Net), with a feature pre-extractor to extract middle-level features, a multi-scale module to fuse features with different receptive fields, and a local attention module based on CBAM [35] to guide the network to focus on local salient features.

2.2 Pooling Operations for Feature Extraction

Feature pooling layers (e.g., average pooling and max pooling) in CNNs serve the dual purpose of providing increasingly abstract representations as well as reducing feature map dimensionality and computational overhead in subsequent convolutional layers. If the convolutional filter size is set to be the entire 2D feature map size, then we have **Global Average Pooling** (**GAP**) and **Global Max Pooling** (**GMP**), which compress each channel of a feature map into one pixel, either the average value or the max value among all pixels in the channel, thus transforming a feature map of dimensions $C \times W \times H$ into a vector of dimensions $C \times 1 \times 1$. Many well-known attention mechanisms, e.g., **Squeeze-and-Excitation Network** (**SENet**) [9], **Bottleneck Attention Module** (**BAM**) [24], **Convolutional Block Attention Module** (**CBAM**) [35], and ECA-Net [33], apply GAP and/or GMP to compress feature maps along both the spatial dimension and the channel

dimension, e.g., the classic work of SENet [9] defines the SE block, an architecture unit that integrates two operations: a squeeze operation that employs GAP to aggregate spatial features into a channel feature, and an excitation operation that learns instance-specific channel weights from the squeezed feature to re-weight each channel. While GAP and GMP are widely used in attention mechanisms, they cause loss of spatial/positional information. Many authors designed more sophisticated pooling operations to better compress feature maps. Zhai et al. [38] propose a pooling strategy with stochastic spatial sampling, where the regular down-sampling is replaced by a stochastic version, which acts as a regularizer by performing implicit data augmentation by introducing distortions in the feature maps. Jin et al. [12] propose a two-stage spatial pooling process: rich descriptor extraction to obtain a set of diverse (global and local) deep descriptors that contain more informative cues than GAP, and information fusion to aid the excitation operation to return more accurate re-weight scores. Behera et al. [3] propose **Context-aware Attentional Pooling** (**CAP**) that captures subtle changes via sub-pixel gradients and learns to attend informative integral regions and their importance in discriminating different subcategories without requiring the bounding box and/or distinguishable part annotations. Oh et al. [22] propose **Background-Aware Pooling** (**BAP**) that focuses on aggregating foreground features inside the bounding boxes using attention maps, to help extract high-quality pseudo-segmentation labels for semantic segmentation. Some authors introduce more spatial priors into pooling operations for further feature enhancement. Schuurmans et al. [27] propose superpixel pooling for semantic segmentation, a flexible and efficient pooling strategy that incorporates spatial prior information. Hou et al. [8] propose Coordinate Attention, which factorizes channel attention into two 1D feature encoding processes that aggregate features along the two spatial directions, respectively. Long-range dependencies can be captured along one spatial direction and precise positional information can be preserved along the other spatial direction. Qin et al. [25] prove that GAP is a special case of **Discrete Cosine Transform** (**DCT**), and propose FcaNet with the multi-spectral attention module, which generalizes the existing channel attention mechanism in the frequency domain. Coordinate Attention [8] and FcaNet [25] form the inspirations for our designs of the ASP and SFP modules.

2.3 Edge-AI Algorithms and Systems

To deploy DNNs on resource-constrained edge devices with limited processing and memory resources [1, 17], it is important to develop effective techniques for improving the computation and memory efficiency of DNN inference. One popular approach is model compression with techniques such as pruning and quantization [5, 19, 21]. Bhardwaj et al. [4] propose a Network of Neural Networks, a distributed IoT learning paradigm that compresses a large pre-trained Teacher deep network into several disjoint compressed Student modules without loss of accuracy. Li et al. [14] propose the Edgent framework, which leverages edge servers for DNN collaborative inference through device-edge synergy by exploiting two design knobs, including DNN partitioning that adaptively partitions computation between device and edge, and DNN right-sizing that further reduces computing latency via early-exiting at an appropriate intermediate DNN layer. Wu et al. [36] propose an edge-computing driven and end-to-end framework to perform tasks of image enhancement and object detection under low-light conditions, consisting of a cloud-based enhancement stage to perform illumination enhancement, and an edge device-based detection stage to detect objects based on informative feature maps from the cloud.

Several authors [10, 11, 23, 29] propose input-specific adaptive inference for improving the computation efficiency of DNN inference, i.e., dynamically adjusting the computational effort depending upon the difficulty of the input data. Panda et al. [23] propose **Conditional Deep Learning** (**CDL**), where features computed by the convolutional layers are first used to identify the difficulty level of input samples, and then conditionally activate the deeper layers of the network in the

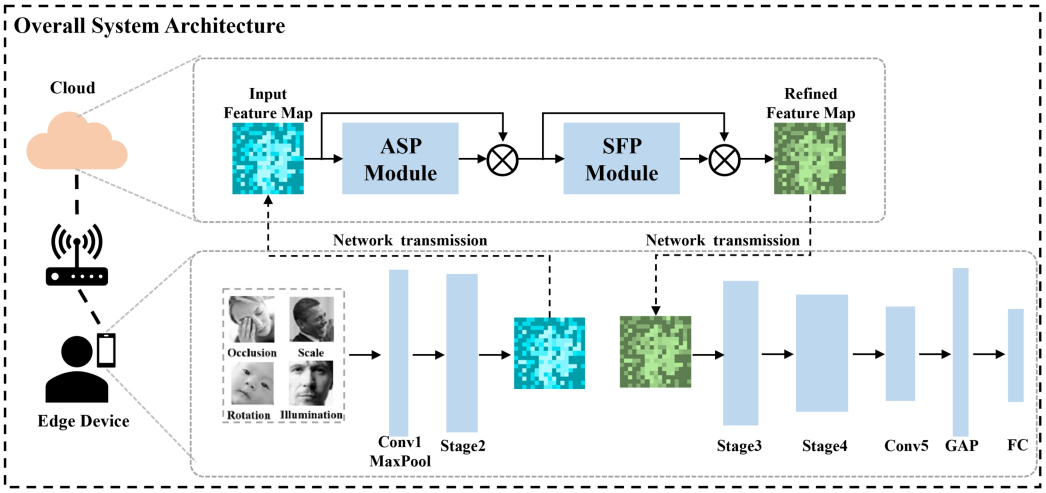


Fig. 1. The overall system architecture consists of a backbone network and optional ASP and SFP modules.

cloud. Jayakodi et al. [10] propose **Learning Energy Accuracy Tradeoff Networks (LEANets)**, which perform input adaptive inferences under multi-objective optimization, thus determining thresholds for different energy and accuracy tradeoffs. Jayakodi et al. [11] propose SETGAN with pre-trained SinGAN structure on remote servers, where its optimal number of scales is carefully determined w.r.t. energy, accuracy, and communication constraints of the mobile device, Stamoulis et al. [29] cast the design of adaptive CNNs as a hyper-parameter optimization problem, where they adopt Bayesian optimization to reach optimal configurations in a few tens of function evaluations. We view this body of work on input-specific adaptive inference as orthogonal and complementary to our work.

3 EDGE-AI-DRIVEN FER FRAMEWORK

In this section, we first present our overall framework in Section 3.1, then the detailed design of the ASP and SFP modules in Sections 3.2 and 3.3.

3.1 Overall Framework

Figure 1 shows our overall system architecture. The backbone network architecture is based on ShuffleNet V2 [20], replacing the original output layer for ImageNet classification. It consists of the following layers: Conv1, MaxPool, Stage2, Stage3, Stage4, Conv5, GAP, and a **Fully-Connected (FC)** layer that outputs the softmax scores of all classes (e.g., seven classes for FER2013 and SFEW datasets, and eight classes for the RaFD dataset in our experiments). The backbone network is deployed on the edge device, and the ASP and SFP modules are deployed in the cloud, connected sequentially. Note that Figure 1 is for illustration purposes only. First, the ASP and SFP modules are not dependent on the specific architecture of the backbone network, and can be used in combination with any other backbone network. Second, the system may contain none, one, or both of the ASP and SFP modules; one or both of them may be offloaded to the cloud; and two modules may be connected in different sequential orders. For example, if none of the two modules is present, then we have the "backbone-only" configuration with a direct connection between Stage 2 and Stage 3.

The ASP module is a spatial attention mechanism that weights each pixel in each channel in the same way; the SFP module is a channel attention mechanism that performs channel-wise scaling,

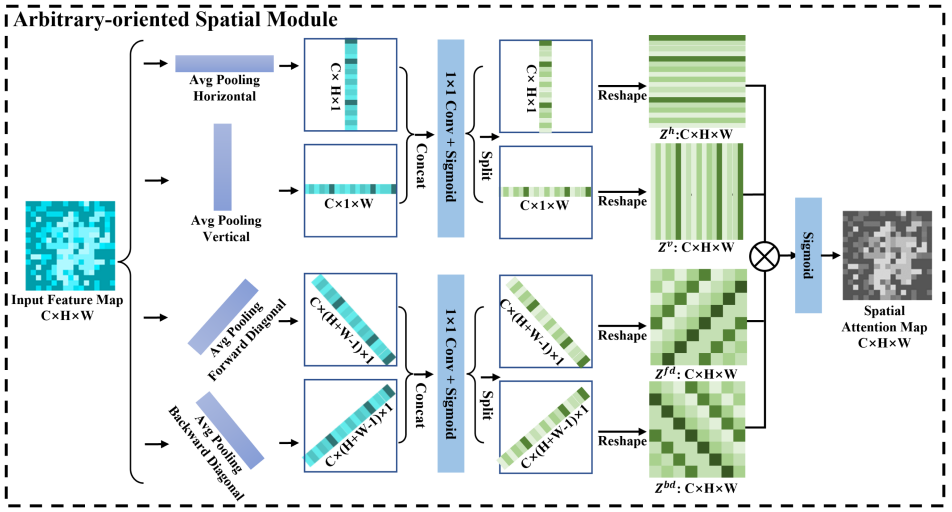


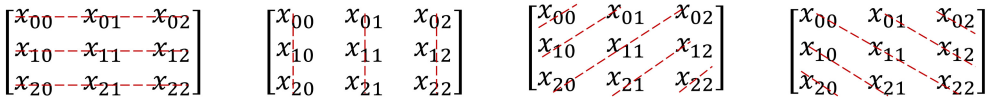
Fig. 2. Structure of the ASP module for spatial attention.

i.e., it weights each pixel of a channel in the same way. Each module outputs an attention map that is multiplied element-wise with the feature map that is used to generate it, to assign different weights to different pixels for spatial attention, or to different channels for channel attention. The FER task has different characteristics from the face recognition task in terms of the most important features. For FER, the important features are the global distribution of the face and low-frequency signal characteristics, whereas, for face recognition, the important features are the color, contour, texture, and other facial features that help distinguish among different faces. The ASP and SFP modules are designed specifically to extract the important features for the FER task. Similar to CBAM [35], the ASP and SFP modules are connected sequentially. Similar to EfficientFace [40], the ASP and SFP modules are added only once after Stage 2 for two reasons: first, deeper feature maps have small spatial sizes, and may not be very helpful for extracting local facial features for FER; second, adding these modules at every stage may cause excessive computation overhead if executed locally on the device, or excessive communication latency if executed remotely in the cloud, which may hinder efficient model deployment on resource-constrained mobile devices.

It is conceivable (although not done in this article) to implement a dynamic adaptive processing strategy based on the architecture in Figure 1, where only the backbone network is executed locally when processing "easy" input images, or when faced with an unreliable network connection to the cloud; the optional attention modules are deployed in the cloud and may be activated on-demand upon detecting difficult inputs or user request to improve both classification accuracy and timing performance.

3.2 Arbitrary-oriented Spatial Pooling (ASP) Module

Figure 2 shows our proposed ASP module. The input feature map has dimensions $C \times H \times W$, where C, H, and W denote the number of channels, height, and width, respectively. We first perform 1D pooling operations in four directions to capture long-range dependencies along them, then apply 1×1 convolution and Sigmoid activation, and finally fuse the four weight maps corresponding to four directions to obtain the final spatial attention map. ASP remedies the shortcomings of SENet [9], which loses spatial/positional information, and of BAM [24] and CBAM [35], which capture local relations with convolution but fail to model long-range dependencies. The ASP module is



(a) Pooling in the horizontal direction.

(b) Pooling in the vertical direction.

(c) Pooling in the forward diagonal direction.

(d) Pooling in the backward diagonal direction.

Fig. 3. Four types of pooling operations in ASP.

inspired by Coordinate Attention [8], but with several differences: (1) We use one 1×1 convolution layer instead of two layers with a reduction followed by an increase in the number of channels; (2) The spatial feature map has dimensions $1 \times H \times W$ to apply the same spatial attention map to all channels, instead of $C \times H \times W$ in Coordinate Attention, since we have a separate SFP module for channel attention.

We illustrate the four types of pooling operations with a toy example. We consider a tiny feature map with dimensions $C \times 3 \times 3$. Equation (1) shows one channel of this feature map. Figure 3 illustrates the four types of pooling operations on each channel of the feature map. (We omit the channel index $0 \le c < C$ for simplicity, with the understanding that the same pooling operation is performed on every channel.)

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ x_{20} & x_{21} & x_{22} \end{bmatrix}. \tag{1}$$

For average pooling in the horizontal direction (Figure 3(a)), we obtain a tensor with dimensions $C \times H \times 1$ by replacing every row with the average value among all the pixels in the row:

$$\begin{bmatrix} z_0^h & z_1^h & z_2^h \end{bmatrix}^T, \tag{2}$$

where each pixel z_i^h is computed as

$$z_0^h = (x_{00} + x_{01} + x_{02})/3, \quad z_1^h = (x_{10} + x_{11} + x_{12})/3, \quad z_2^h = (x_{20} + x_{21} + x_{22})/3. \tag{3}$$

For average pooling in the vertical direction (Figure 3(b)), we obtain a tensor with dimensions $C \times 1 \times W$ by replacing every column with the average value among all the pixels in the column:

$$\begin{bmatrix} z_0^v & z_1^v & z_2^v \end{bmatrix}, \tag{4}$$

where each pixel z_j^v is computed as

$$z_0^v = (x_{00} + x_{10} + x_{20})/3, \quad z_1^v = (x_{01} + x_{11} + x_{21})/3, \quad z_2^v = (x_{02} + x_{12} + x_{22})/3. \tag{5}$$

For average pooling in the forward diagonal direction (Figure 3(c)), we obtain a tensor with dimensions $C \times (H + W - 1) \times 1$ by computing the average value along all the pixels along each of the $H + W - 1 = 5$ forward diagonal lines with different offsets:

$$\begin{bmatrix} z^{fd}(0) & z^{fd}(1) & z^{fd}(2) & z^{fd}(3) & z^{fd}(4) \end{bmatrix}^T, \tag{6}$$

where each pixel $z^{fd}(i)$ is computed as

$$\begin{aligned} &z_0^{fd} = x_{00}, && z_1^{fd} = (x_{10} + x_{01})/2, \quad z_2^{fd} = (x_{20} + x_{11} + x_{02})/3, \\ &z_3^{fd} = (x_{21} + x_{12})/2, && z_4^{fd} = x_{22}. \end{aligned} \tag{7}$$

For average pooling in the backward diagonal direction (Figure 3(d)), we obtain a tensor with dimensions $C \times (H + W - 1) \times 1$ by computing the average value along all the pixels along each of the $H + W - 1 = 5$ backward diagonal lines with different offsets:

$$\left[z^{bd}(0) \quad z^{bd}(1) \quad z^{bd}(2) \quad z^{bd}(3) \quad z^{bd}(4) \right]^T, \tag{8}$$

where each pixel $z^{bd}(i)$ is computed as

$$\begin{aligned} &z_0^{bd} = x_{02}, && z_1^{bd} = (x_{01} + x_{12})/2, && z_2^{bd} = (x_{00} + x_{11} + x_{22})/3, \\ &z_3^{bd} = (x_{10} + x_{21})/2, && z_4^{bd} = x_{20}. \end{aligned} \tag{9}$$

For the two tensors in Equations (2) and (4) from horizontal and vertical pooling, we perform the following steps:

— Concatenate them to form a tensor with dimensions $C \times (H + W) \times 1$.
— Apply a 1×1 convolution layer with output dimensions $1 \times (H + W) \times 1$ (with one output channel), followed by the Sigmoid activation function.
— Split the output tensor with dimensions $1 \times (H + W) \times 1$ into two tensors of dimensions $1 \times H \times 1$ and $1 \times 1 \times W$.
— Reshape each tensor into the same spatial dimensions of the input feature map to obtain two weight maps Z^h and Z^v with dimensions $1 \times H \times W$, by setting each x_{ij} to its corresponding average value after pooling in Equations (3) and (5), e.g., $x_{00} = x_{01} = x_{02} = z_0^h$. (This reshape operation is used to achieve the same effect as NumPy's broadcasting operation when multiplying a vector with a matrix [30].)

For the two tensors in Equations (6) and (8) from forward and backward diagonal pooling, we perform the following steps:

— Concatenate them to form a tensor with dimensions $C \times 2(H + W - 1) \times 1$.
— Apply a 1×1 convolution layer with output dimensions $1 \times 2(H + W - 1) \times 1$ (with one output channel), followed by the Sigmoid activation function.
— Split the output tensor with dimensions $1 \times 2(H + W - 1) \times 1$ into two tensors of the same dimensions $1 \times (H + W - 1) \times 1$.
— Reshape each tensor into the same spatial dimensions of the input feature map to obtain two weight maps Z^{fd} and Z^{rd} with dimensions $1 \times H \times W$, by setting each x_{ij} to its corresponding average value after pooling in Equations (7) and (9), e.g., $x_{10} = x_{01} = z_1^{fd}$. (This reshape operation has no equivalence in NumPy.)

The final spatial attention map with dimensions $1 \times H \times W$ is computed with element-wise multiplication of the four weight maps, i.e., $Z^h \otimes Z^v \otimes Z^{fd} \otimes Z^{bd}$.

3.3 Scalable Frequency Pooling (SFP) Module

DWT decomposes a signal into a set of mutually orthogonal wavelet basis functions. These functions differ from sinusoidal basis functions of DCT in that they are spatially localized—that is, nonzero over only part of the total signal length. In addition, DWT has the multi-scale property that helps to address the scale variations. Figure 4 shows multiple levels of 2D-DWT applied to an input **High-Resolution** (**HR**) image. After each level of 2D-DWT, the input is decomposed into four components LL, LH, HL, and HH, where L denotes Low-frequency and H denotes High-frequency along either the horizontal or vertical direction, e.g., LL denotes the low-frequency components along both directions. After three levels of 2D-DWT, we obtain three components LL1, LL2, and LL3 with increasingly lower frequencies. The motivation is that the low-frequency

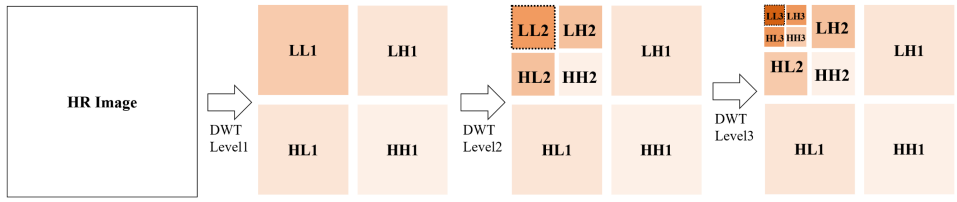


Fig. 4. Multiple levels of 2D DWT applied to an input HR image.

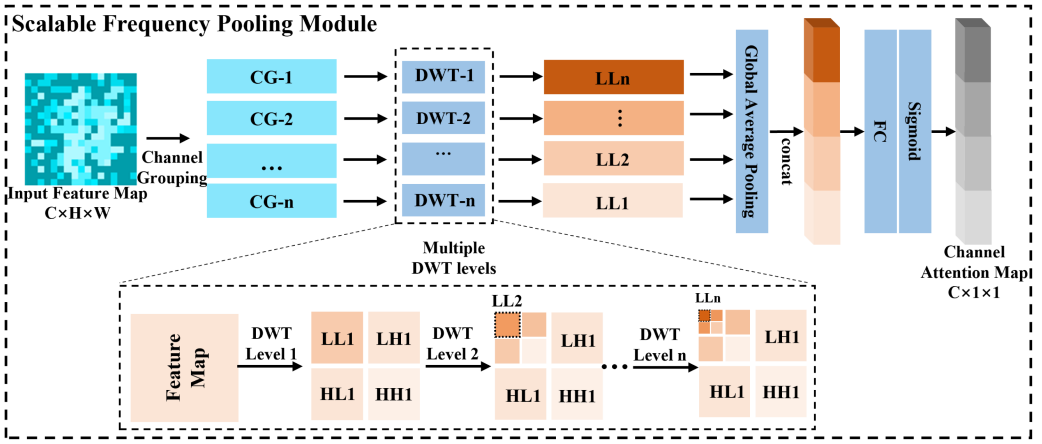


Fig. 5. Structure of the SFP module for channel attention.

components of an image contain the dominant information for FER, e.g., a human emotion may be recognizable with low-frequency components only.

Figure 5 shows the detailed design of SFP based on DWT (it is inspired by FcaNet [25], which is based on DCT). Consider an input feature map X with dimensions $C \times H \times W$. We assume $C = 8$ as an example to illustrate the operation of the SFP module. We group the C channels into n **Channel Groups (CGs)** with the PyTorch function `torch.chunk`, e.g., with $C = 8$ channels to be grouped into $n = 3$ CGs, we obtain three CGs: CG-1 contains three channels with indices 0–2, CG-2 contains three channels with indices 3–5, and CG–3 contains two channels with indices 6–7. We apply DWT Level-i to each channel group CG-i to obtain the different frequency components, and only keep the LL component and discard the rest of LH, HL, and HH components. For example, we keep the LL1 component for CG-1 as a feature map with dimensions $3 \times H/2 \times W/2$, the LL2 component for CG-2 as a feature map with dimensions $3 \times H/4 \times W/4$, and the LL3 component as a feature map for CG-3 with dimensions $2 \times H/8 \times W/8$. We then apply GAP to compress each feature map into a single value for each channel, resulting in three vectors with dimensions $3 \times 1 \times 1$, $3 \times 1 \times 1$, $2 \times 1 \times 1$ for CG-1, CG-2, and CG-3, respectively. We then concatenate them along the channel dimension to obtain a vector with dimensions $8 \times 1 \times 1$. We then apply a FC layer with the same input/output dimensions, instead of two FC layers with a bottleneck hidden layer as in CBAM [35] and SENet [9]. (This is inspired by ECA-Net [33], which empirically showed that the SE block [9] that employs one single FC layer may work better than two FC layers with dimensionality reduction in the bottleneck hidden layer.) Finally, we use the Sigmoid activation function to scale each vector element into the range of $[0, 1]$ as the channel attention weights.

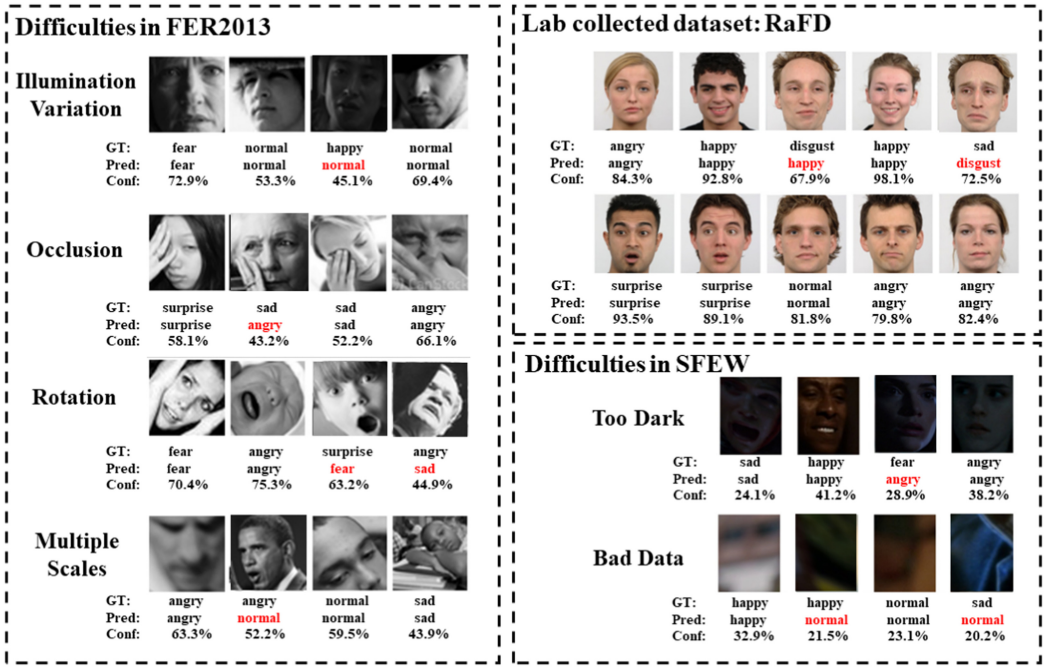


Fig. 6. Selected samples from the three datasets: FER2013, RaFD, and SFEW.

4 PERFORMANCE EVALUATION

In this section, we present the experimental setup in Section 4.1; evaluation of classification accuracy in Section 4.2; ablation studies to evaluate different design choices in Section 4.3, and evaluation of timing performance in Section 4.4.

4.1 Experimental Setup

We set up our experiments in a room with a Wi-Fi connection with maximum network bandwidth of 1 Mb/s. The Cloud[1] is simulated by a server machine with 2.1 GHz E5-2620 processor and 10 GB Memory, a Titan 1080Ti GPU for both training and inference in the Cloud. We consider two types of edge devices: a laptop computer and a low-end smartphone (Redmi K30). The laptop computer contains an Intel i7-9750h CPU with 6 cores, with base frequency 2.60 GHz and maximum Turbo Frequency 4.50 GHz, and 8 GB memory. The smartphone contains a Qualcomm Snapdragon 730G SoC with 8 cores (2×2.2 GHz and 6×1.8 GHz), and 6 GB memory.

We consider three well-known FER datasets: a laboratory-controlled dataset **Radboud Faces Database (RaFD)** [13], and two FER in the wild datasets FER2013 [2] and **Static Facial Expressions in the Wild (SFEW)** [6].

— The RaFD dataset contains 4,824 images collected from 67 participants. Each participant makes eight facial expressions in three different gaze directions, which are captured from three different angles, with eight expression labels (Anger, Disgust, Fear, Happiness, Sadness, Surprise, Contempt, and Neutrality).

[1]The Cloud in our experimental setup may be more suitably called an edge server since our experimental setup is based on a local area network. Nonetheless, we use the term Cloud to refer to a remote server in general, and control the Wi-Fi network bandwidth to simulate a low-bandwidth wide area network.

Table 1. Classification Accuracy for Different Datasets (%)

Dataset	ResNet18	ResNet50	MobileNetV2	EfficientFace	MA-Net	RAN	Ours
RAFD	84.59	85.16	83.86	85.21	85.27	85.42	**85.57**
FER2013	58.72	60.45	57.12	62.43	61.97	61.52	**63.26**
SFEW	55.40	56.58	54.19	56.54	**59.40**	54.19	56.81

Bold indicates best performance.

— The FER2013 dataset is a large-scale unconstrained dataset collected by Google image search. All images are registered and resized to 48*48 pixels after rejecting incorrectly labeled frames and adjusting the cropped region. FER2013 contains a training set (28,709 images), a validation set (3,589 images), and a test set (3,589 images), with seven expression labels (Anger, Disgust, Fear, Happy, Sad, Surprised, and Neutral).
— The SFEW dataset contains a training set (958 images), a validation set (436 images), and a test set (372 images), with seven expression labels (Neutral, Happiness, Sadness, Surprise, Fear, Disgust, and Anger). SFEW contains difficult images for FER in the wild, with varied head poses, large age range, occlusions, varied focus, different resolution of face and close to real-world illumination.

Figure 6 shows selected samples from the three datasets, to provide some intuition about their appearances and levels of difficulty. Under each sample, we show the **Ground Truth** (**GT**), the predicted label (Pred) from our model, i.e., the full model of (backbone+ASP+SFP), with the highest confidence score, and the confidence score (Conf) of the prediction.

For all datasets, we resize all the input face images into the same dimensions of $3 \times 224 \times 224$ with the PyTorch function `Resize`, i.e, an RGB color image with a spatial size of 224×224 pixels. We perform data augmentation with random cropping and random horizontal flipping to avoid over-fitting. The model weights are initialized randomly. The model is pre-trained on the largest FER2013 dataset for 100 epochs, with the SGD optimizer, mini-batch size of 128, and initial learning rate of 0.1, which is reduced to 1/10 of its previous value every 30 epochs. For the RaFD and SFEW datasets, we start from this pre-trained model, and replace its output layer according to the corresponding set of classes for each dataset, initialized with random weights. The model is then fine-tuned on either RaFD or SFEW for 100 epochs with a constant learning rate of 0.001.

4.2 Evaluation of Classification Accuracy

We first evaluate the classification accuracy metric for the three datasets. We consider the full model of (backbone+ASP+SFP), and Daubechies wavelet with three DWT levels in the SFP module. (The choice of these configurations is supported by ablation studies in Section 4.3.) We consider the following comparison baselines: ResNet18 and ResNet50 [7], ResNet models with different numbers of layers (18 and 50 deep layers, respectively); MobileNetV2 [26], a lightweight DNN for mobile devices; EfficientFace [40], a lightweight DNN designed for FER; MA-Net [39] and RAN (with ResNet18 backbone) [32], two relatively heavyweight DNNs designed for FER. As shown in Table 1, for both RaFD and FER2013, our model achieves the highest accuracy, with a larger winning margin over the baselines for FER2013 than for RaFD, since our model is designed for FER in the wild. For the SFEW dataset, our model obtains the second-highest accuracy of 56.81%. Although it does not outperform the winner MA-Net, our model is much more lightweight than MA-Net (c.f. Table 4). We conclude that our model strikes a good balance between accuracy and computation efficiency.

In the next two subsections, we consider the FER2013 dataset only for ablation studies and timing measurements.

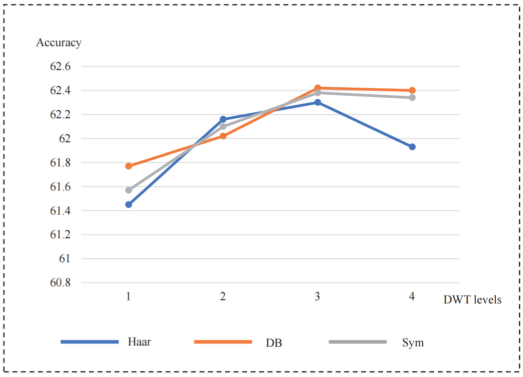


Fig. 7. Ablation study with different DWT bases and levels in the SFP module.

Table 2. Ablation Study with Different Combinations of the ASP and SFP Modules

Model	Accuracy (%)
backbone only	60.25
backbone+ASP	61.88
backbone+SFP	61.34
backbone+ASP+SFP	**63.26**
backbone+SFP+ASP	62.58

Bold indicates best performance.

4.3 Ablation Studies

Different DWT bases and levels in the SFP module. In this ablation study, we consider the configuration of (backbone+SFP) without the ASP module and consider three different DWT bases: HAAR, **Daubechies** (**DB**) and **Symlets** (**SYM**), and different numbers of DWT Levels in the SFP module. As shown in Figure 7, for all three bases, the classification accuracy increases with DWT level going from 1 to 3, and stays the same or drops slightly with DWT level going from 3 to 4. Daubechies wavelet achieves the highest accuracy with DWT level 3. Thus, we determine the optimal configuration of DWT in the SFP module to be Daubechies wavelet with three DWT levels, which is used in the following experiments.

Different configurations of the ASP and SFP modules. Table 2 shows the ablation study with none, one, or both of the ASP and SFP modules. If both modules are present, we consider two sequential orders: either ASP preceding SFP (backbone+ASP+SFP) or SFP preceding ASP (backbone+SFP+ASP). We can see that the best-performing configuration is (backbone+ASP+SFP) with an accuracy of 63.26%.

Different pooling directions in the ASP module. Table 3 shows the ablation study with different combinations of pooling directions in the ASP module, for the configuration of (backbone+ASP), without the SFP module. We can see that the best-performing alternative is pooling in all four directions, with the highest accuracy of 61.88%. Pooling in any combination of the four directions helps to increase accuracy to different degrees. Pooling in the horizontal and/or vertical directions is generally more effective than pooling in the forward and/or backward diagonal directions. The reason may be that the conventional frontal and upright head pose accounts for a large proportion of all images in FER2013, even though it contains many face images with different head poses.

Table 3. Ablation Study with Different Combinations of Pooling Directions in the ASP Module

Pooling Direction	Accuracy (%)
backbone only	60.25
H	61.43
V	60.98
H+V	61.57
FD	60.72
BD	60.93
FD+BD	61.02
H+V+FD+BD	61.88

Abbreviations: H for Horizontal; V for Vertical; FD for Forward Diagonal; BD for Backward Diagonal.

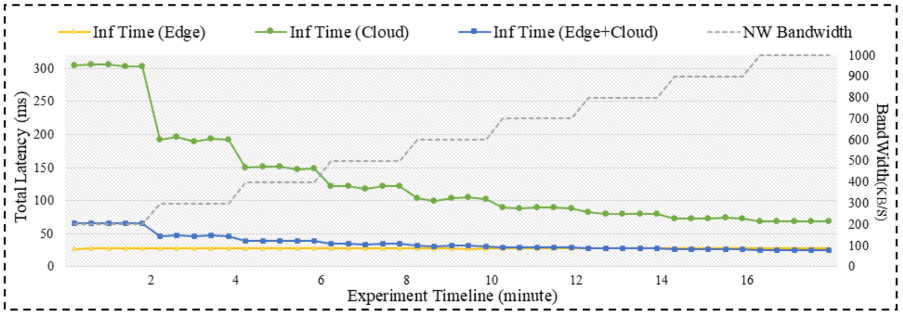
Table 4. DNN Model Size Measured by Number of Parameters, Computation Load **Measured by Multiply-Accumulate Operations** (**MACs**), and Inference time on Two Edge Devices

Model	Params (M)	MACs (G)	Inf Time on Laptop (ms)	Inf Time on Phone (ms)
ResNet18	11.18	1.82	114.8	249.3
ResNet50	23.52	4.12	239.1	510.8
MobileNetV2	2.23	0.31	29.3	93.7
EfficientFace	1.28	0.15	12.9	32.7
MA-Net	42.29	1.89	139.0	354.2
RAN	11.19	4.52	255.7	595.1
Ours (backbone only)	1.26	0.15	11.6	25.4
Ours (full model)	1.91	0.29	27.6	62.9

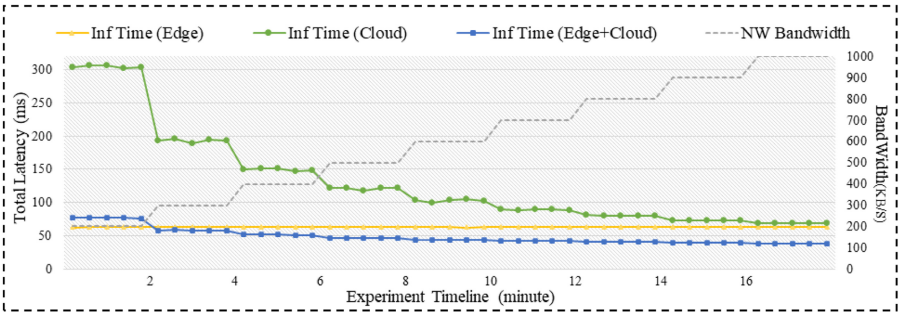
4.4 Evaluation of Timing Performance

Table 4 compares model size, computation load, and inference time on different edge devices. The number of parameters and computation load in MACs are obtained with the open-source Flops counter tool for CNNs [28]. The DNN inference time is roughly proportional to the MACs, but not exactly due to the different number of parameters and memory access patterns of different DNN architectures. The row labeled "Ours (full model)" refers to the full model with (backbone+ASP+SFP). We can see that our model is more lightweight than most comparison baselines, except EfficientFace. We can also see that the ASP and SFP modules consume a significant proportion of the total DNN inference time, esp. on the resource-constrained smartphone platform, which motivates the need for partial offloading of these modules from resource-constrained edge devices to the cloud.

Each input image has dimensions $3 \times 224 \times 224$ with size 150.5 KB (each pixel is encoded by 1 Byte). The intermediate feature map as output from Stage 2 in Figure 1 has dimensions $8 \times 28 \times 28$ with size 6.3 KB. During one forward DNN inference, the total data size for transmitting the intermediate feature maps to and from the cloud is $6.3 * 2 = 12.6$ KB, taking into account two-way data transmission between the edge device and the cloud. The total data size for transmitting one image to the cloud and a classification result back to the device (which has negligible data size



(a) A laptop computer as the edge device.



(b) A smartphone as the edge device.

Fig. 8. Timing performance measurements with varying network bandwidth with two types of edge devices.

but not latency) is 150.5 KB. This highlights the benefits of transmitting the intermediate feature maps instead of the image pixels in terms of reduced data size and hence network transmission latency.

Next, we measure the DNN inference latency with varying network bandwidth with two types of edge devices, considering three offloading strategies:

— No offloading (Edge): the full model (backbone+ASP+SFP) runs locally on the edge device.
— Full offloading (Cloud): the edge device sends the input image to the cloud, which runs the full model (backbone+ASP+SFP), and returns the classification result to the edge device.
— Partial offloading (Edge+Cloud): the backbone runs on the edge device, and sends the intermediate feature map to the cloud, which hosts the ASP and SFP modules (as shown in Figure 1).

We use the open-source NetLimiter tool[2] to control the Wi-Fi network bandwidth and increase it stepwise from 200 KB/s to 1 MB/s by 100 KB/s every 2 minutes. The edge device processes the same image in an infinite loop during the entire experiment process. Figure 8 plots the total latency of each of the three offloading strategies. We make the following observations:

— Both the full offloading and the partial offloading strategies benefit from increased network bandwidth: the higher the network bandwidth, the lower the total latency due to reduced network transmission latency.

[2]https://www.netlimiter.com.

— With the laptop computer as the edge device, the best strategy that achieves the lowest latency is no offloading, i.e., the full model runs locally on the laptop, thanks to the powerful computing capability of the laptop computer.
— With the more resource-constrained smartphone as the edge device, the best strategy that achieves the lowest latency is partial offloading when the network bandwidth exceeds 300 KB/s. (Full or partial offloading is expected to be even more beneficial for more resource-constrained edge devices such as smart cameras.)

5 CONCLUSIONS

In this article, we present an Edge-AI-driven framework for accurate and efficient FER in the wild. First, we propose two attention modules, ASP and SFP, that can be used in combination with any backbone network, to encode both multi-direction spatial information and multi-scale frequency information for effective feature extraction in the presence of realistic occlusions, illumination, scale, and head pose variations. Second, we propose an edge-cloud joint inference architecture for FER to achieve low-latency inference, with partial offloading of the two optional attention modules to the cloud. Performance evaluation with several FER datasets demonstrates the effectiveness of our proposed method in terms of both accuracy and computation efficiency.

REFERENCES

[1] Zaid Al-bayati, Qingling Zhao, Ahmed Youssef, Haibo Zeng, and Zonghua Gu. 2015. Enhanced partitioned scheduling of mixed-criticality systems on multicore platforms. In *Proceedings of the 20th Asia and South Pacific Design Automation Conference*. IEEE, 630–635.
[2] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. 2016. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the ACM International Conference on Multimodal Interaction*. 279–283.
[3] Ardhendu Behera, Zachary Wharton, Pradeep R. P. G. Hewage, and Asish Bera. 2021. Context-aware attentional pooling (CAP) for fine-grained visual classification. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 929–937.
[4] Kartikeya Bhardwaj, Chingyi Lin, Anderson L. Sartor, and Radu Marculescu. 2019. Memory- and communication-aware model compression for distributed deep learning inference on IoT. *ACM Transactions on Embedded Computing Systems* 18, 5s (2019), 82:1–82:22.
[5] Tejalal Choudhary, Vipul Kumar Mishra, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review* 53, 7 (2020), 5113–5155.
[6] Abhinav Dhall, Roland Goecke, Simon Lucey, and Tom Gedeon. 2011. Static facial expression analysis in tough conditions: Data, evaluation protocol, and benchmark. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2106–2112.
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[8] Qibin Hou, Daquan Zhou, and Jiashi Feng. 2021. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 13713–13722.
[9] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7132–7141.
[10] Nitthilan Kanappan Jayakodi, Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. 2020. Design and optimization of energy-accuracy tradeoff networks for mobile platforms via pretrained deep models. *ACM Transactions on Embedded Computing Systems* 19, 1 (2020), 4:1–4:24.
[11] Nitthilan Kanappan Jayakodi, Janardhan Rao Doppa, and Partha Pratim Pande. 2020. SETGAN: Scale and energy tradeoff GANs for image applications on mobile platforms. In *Proceedings of the IEEE/ACM International Conference On Computer Aided Design*. 23:1–23:9.
[12] Xin Jin, Yanping Xie, Xiu-Shen Wei, Borui Zhao, Zhao-Min Chen, and Xiaoyang Tan. 2022. Delving deep into spatial pooling for squeeze-and-excitation networks. *Pattern Recognition* 121 (2022), 108159.
[13] Oliver Langner, Ron Dotsch, Gijsbert Bijlstra, Daniel H. J. Wigboldus, Skyler T. Hawk, and A. D. Van Knippenberg. 2010. Presentation and validation of the Radboud faces database. *Cognition and Emotion* 24, 8 (2010), 1377–1388.

[14] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. 2020. Edge AI: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications* 19, 1 (2020), 447–457.
[15] Yong Li, Jiabei Zeng, Shiguang Shan, and Xilin Chen. 2018. Patch-gated CNN for occlusion-aware facial expression recognition. In *Proceedings of the International Conference on Pattern Recognition*. 2209–2214.
[16] Yong Li, Jiabei Zeng, Shiguang Shan, and Xilin Chen. 2019. Occlusion aware facial expression recognition using CNN with attention mechanism. *IEEE Transactions on Image Processing* 28, 5 (2019), 2439–2450.
[17] Guangdong Liu, Ying Lu, Shige Wang, and Zonghua Gu. 2014. Partitioned multiprocessor scheduling of mixed-criticality parallel jobs. In *Proceedings of the 2014 IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 1–10.
[18] Yuanyuan Liu, Xiaohui Yuan, Xi Gong, Zhong Xie, Fang Fang, and Zhongwen Luo. 2018. Conditional convolution neural network enhanced random forest for facial expression recognition. *Pattern Recognition* 84 (2018), 251–261.
[19] Siyu Luan, Zonghua Gu, Rui Xu, Qingling Zhao, and Gang Chen. 2023. LRP-based network pruning and policy distillation of robust and non-robust DRL agents for embedded systems. *Concurrency and Computation: Practice and Experience* (2023), e7351.
[20] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. 2018. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In *Proceedings of the European Conference on Computer Vision*. 122–138.
[21] Wenjia Meng, Zonghua Gu, Ming Zhang, and Zhaohui Wu. 2017. Two-bit networks for deep learning on resource-constrained embedded devices. CoRR abs/1701.00485 (2017).
[22] Youngmin Oh, Beomjun Kim, and Bumsub Ham. 2021. Background-aware pooling and noise-aware loss for weakly-supervised semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6913–6922.
[23] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. 2016. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition*. 475–480.
[24] Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. 2018. Bam: Bottleneck attention module. In *Proceedings of British Machine Vision Conference*. 147–160.
[25] Zequn Qin, Pengyi Zhang, Fei Wu, and Xi Li. 2021. Fcanet: Frequency channel attention networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 783–792.
[26] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the CVPR*. 4510–4520.
[27] Mathijs Schuurmans, Maxim Berman, and Matthew B. Blaschko. 2018. Efficient semantic image segmentation with superpixel pooling. CoRR abs/1806.02705 (2018).
[28] Vladislav Sovrasov. 2022. Ptflops: A flops counting tool for neural networks in pytorch framework. https://github.com/sovrasov/flops-counter.pytorch.
[29] Dimitrios Stamoulis, Ting-Wu (Rudy) Chin, Anand Krishnan Prakash, Haocheng Fang, Sribhuvan Sajja, Mitchell Bognar, and Diana Marculescu. 2018. Designing adaptive neural networks for energy-constrained image classification. In *Proceedings of the International Conference on Computer-Aided Design*. 23.
[30] Stefan Van Der Walt, S. Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering* 13, 2 (2011), 22–30.
[31] Noranart Vesdapunt and Baoyuan Wang. 2021. CRFace: Confidence ranker for model-agnostic face detection refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1674–1684.
[32] Kai Wang, Xiaojiang Peng, Jianfei Yang, Debin Meng, and Yu Qiao. 2020. Region attention networks for pose and occlusion robust facial expression recognition. *IEEE Transactions on Image Processing* 29 (2020), 4057–4069.
[33] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. 2020. ECA-Net: Efficient channel attention for deep convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11534–11542.
[34] Wenjing Wang, Wenhan Yang, and Jiaying Liu. 2021. HLA-Face: Joint high-low adaptation for low light face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 16195–16204.
[35] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. CBAM: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision*. Vol. 11211. 3–19.
[36] Yirui Wu, Haifeng Guo, Chinmay Chakraborty, Mohammad Khosravi, Stefano Berretti, and Shaohua Wan. 2023. Edge computing driven low-light image dynamic enhancement for object detection. *IEEE Transactions on Network Science and Engineering* (2023). DOI: 10.1109/TNSE.2022.3151502
[37] Siyue Xie, Haifeng Hu, and Yongbo Wu. 2019. Deep multi-path convolutional neural network joint with salient region attention for facial expression recognition. *Pattern Recognition* 92 (2019), 177–191.

[38] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogério Schmidt Feris. 2017. S3Pool: Pooling with stochastic spatial sampling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. 4003–4011.

[39] Zengqun Zhao, Qingshan Liu, and Shanmin Wang. 2021. Learning deep global multi-scale and local attention features for facial expression recognition in the wild. *IEEE Transactions on Image Processing* 30 (2021), 6544–6556.

[40] Zengqun Zhao, Qingshan Liu, and Feng Zhou. 2021. Robust lightweight facial expression recognition network with label distribution training. In *Proceedings of the AAAI*. 3510–3519.

Received 6 May 2022; revised 12 December 2022; accepted 17 February 2023