IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 14, 2021, accepted June 29, 2021, date of publication July 9, 2021, date of current version July 23, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095844

Mobile-Optimized Facial Expression Recognition Techniques

NIZAR EL ZARIF[1], LEILA MONTAZERI[1], FRANÇOIS LEDUC-PRIMEAU[1], (Member, IEEE), AND MOHAMAD SAWAN[1,2], (Fellow, IEEE)

[1]Department of Electrical Engineering, Polytechnique Montreal, Montréal, QC H3T 1J4, Canada
[2]CenBRAIN, School of Engineering, Westlake University, Hangzhou, Zhejiang 310024, China

Corresponding author: Nizar El Zarif (nizar.el-zarif@polymtl.ca)

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and in part by CMC Microsystems.

ABSTRACT This paper presents two novel facial expression recognition techniques: the real-time ensemble for facial expression recognition (REFER) and the facial expression recognition network (FERNet). Both approaches can detect facial expressions from various poses, distances, angles, and resolutions, and both techniques exhibit high computational efficiency and portability. REFER outperforms the existing approaches in terms of cross-dataset accuracy, making it an ideal network to use on fresh data. FERNet is a compact convolutional neural network that uses both geometric and texture features to achieve up to 98% accuracy on the MUG dataset. Both approaches can process 14 frames per second (FPS) from a live video capture on a battery-powered Raspberry Pi 4.

INDEX TERMS Facial expression recognition, machine learning, multithreaded, active shape model, pose-invariant.

I. INTRODUCTION

There are seven essential and recognizable facial expressions that do not require translators and can be understood across cultures and languages. These seven emotions are anger, neutral, fear, disgust, happiness, sadness, and surprise. Even people who are blind from birth can express the correct facial expressions without ever seeing them [1]. While facial expressions can be understood relatively easily by humans, they are challenging for machines because a facial expression detection algorithm must be sensitive to small variations in the face while being robust to changing environmental conditions. In addition, humans tend to express emotions slightly differently, which makes the problem even more difficult. While other senses can be used to estimate emotions, such as hearing, sight is the most accurate indicator of emotion.

Automatic facial expression detection has many beneficial uses in human-computer interactions [2]. Its applications extend to the gaming industry, commerce, medical field, academics, or even personal use. For example, horror-based video games can provide challenges, such as spawning new enemies, based on the player's fear level, which is dynamically extracted from his/her facial expressions. In commerce, facial expression detection can also measure people's reactions to movies, food, advertisements, and clothing. In the medical field, facial expression detection can help in personalized medicine, such as pain measurement [3], or rehabilitation monitoring. It can also provide personalized stimulation to complement a system that emulates retinal processing for people with low vision [4]. In education, facial expression detection can aid teachers by measuring the engagement of online students in virtual classrooms. For personal use, facial expression detection can be extended to tailor music to listeners' emotions or measure the drowsiness or responsiveness of drivers to ensure their safety [5].

For facial expression recognition there are two high-level approaches. The traditional approach relies on feature extraction using specific descriptors. This can be either texture, geometric, hybrid, or generic features. Geometric features, such as [6]–[10] have lower memory and computational requirements, which makes them ideal for real-time applications. Texture features are significantly more computationally expensive than geometric features, but they provide a high accuracy. Some of the more popular approaches can be seen in [11]–[20]. Generic features [21], [22] use general-purpose descriptors. They are relatively easy to implement and use but are not as accurate or as fast as the geometric or texture approaches. Hybrid features [23] are less common since they require both geometry and texture at the same time, which increases the complexity. The second high-level approach to

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

facial expression recognition is to use deep neural networks to skip feature extraction, since neural networks will learn the features from data, as seen in [24]–[51]. This paper proposes two approaches to facial expression recognition. The first is based on the traditional geometric approach with a custom scale and poses invariant descriptors with a fast implementation to ensure the best accuracy on fresh data with a high real-time throughput. The other approach is a neural-network approach with a simple feature extraction that ensures a high accuracy on familiar data with a very fast and low latency execution pipeline, which is ideal for real-time embedded applications.

The remainder of the article is organized as follows. We first review related works in Section II.

Then, Section III introduces the real-time ensemble for facial expression recognition (REFER) technique, and Section IV introduces the FERNet convolutional neural network approach. Section V presents the accuracy and complexity results for the two proposed algorithms and compares them to the state of the art. Finally, Section VI concludes the paper.

II. RELATED WORK

There are three steps in any facial expression recognition (FER) task: feature extraction, training, and testing. The most crucial part of traditional approaches in FER is feature extraction, as it determines which image points are the most distinct for classification while being robust to variations in the scene. A deep neural network can skip feature extraction since the hidden layers in neural networks can detect abstract features.

A. GEOMETRIC FEATURES

The geometric features are based on the face's geometry. Some of these features include the distances and angles between different facial traits, the distance between the mouth and the chin, and the curvature of the mouth. Advances in computer vision have allowed for the creation of the active shape model (ASM) and the active appearance model (AAM) [6]. In both approaches, several points across the face are tracked in real time with a low overhead, such as in [7], [8]. The greatest advantage of the geometric features is that they tend to be computationally inexpensive and memory-efficient due to their limited number, which is important for real-time and embedded applications. However, they are sensitive to scale and face orientation, which makes them prone to registration errors [9]. Liu *et al.* [10] showed that using a subset of the features can improve the detection for one dataset, but at the cost of others. The two proposed approaches solve the issue of both orientation and scale, and greatly improve the detection of geometric features by performing scaling on distances in real time.

B. TEXTURE FEATURES

The second type of feature extraction is based on texture features, which use pixel intensity information of the face to classify expressions. These features can capture smaller face details, such as wrinkles and curves. Texture features tend to require considerable memory due to the need to store and process a large array of pixel data. In addition, textures are illumination-dependent; hence, they suffer from uneven lighting. The latter can be solved by using local texture features instead of global features [11], [12]. Local binary patterns [13] or Gabor wavelets [14] are two of the more popular approaches for texture-based feature extraction, but the latter can be memory and computationally expensive. In [15], the authors used a principal component analysis and template matching to achieve an accuracy that surpasses 99% on personal images. However, with no cross-validation and limited data, the method can be prone to overfitting and requires previous facial and emotional knowledge. To compute the local energy and distance based on the Symlet wavelet with optical flow, researchers in [16] created an unsupervised learning technique based on active contouring and moving features, which obtained an accuracy of 87% on the extended Cohn-Kanade (CK+) dataset [17]. In [18], the authors extracted the region around the nose and the eyes and used the LBP feature with weighted sum voting. They obtained up to a 90% accuracy on the CK+ dataset and 78% on the multimedia understanding group (MUG) dataset [52]. Generic features such as SURF have been used to describe facial features and perform FER with a good accuracy of up to 96% on the MUG dataset [21], [22]. 2D linear discriminant analysis can also obtain a good detection rate [20],

C. HYBRID FEATURES

The third approach to feature extraction is based on hybrid models, using both geometric and texture features. This approach should give more accurate results than each feature alone. In [23], the authors used hybrid features to obtain an accuracy between 85% and 95% using the JAFFE and CK+ datasets. However, execution speed is a considerable drawback to this type of descriptor, as it requires storing and processing both texture and geometric features and then efficiently combining them to produce accurate results.

D. NEURAL NETWORKS

Neural networks are rapidly growing in popularity in machine learning since Alexnet won the ImageNet competition in 2012 [24], [25]. Since then, many neural network-based approaches have been introduced.

The authors of [26] proposed a mixture of geometric and texture features by training both SVM and neural networks. Their results showed a similar performance; however, neural networks tend to overfit the model. As a result, neural networks can only classify the input dataset with a high accuracy, but will underperform with new types of data. Similarly, the authors of [27] used a convolutional neural network (CNN) for facial expression detection, with a high accuracy, but their network underperforms during cross-validation tests, which indicates overfitting. Recently, Sen *et al.* used angles and LBP as geometric and texture features to achieve

accuracies of 78% and 91.85% using the MUG and CK+ datasets, respectively [28].

The authors of [29] used a CNN along with the ensemble method of voting to build and train the classifier. It resulted in a decent EmotiW challenge result with an average of 61.29% of the test set's detection. Researchers in [30] introduced a neural network for facial expression detection based on the difference in the geometric features and the difference in consecutive frames. They then combined both the temporal and geometric features with an accuracy rate of 45.5% on the CK+ dataset. Similarly, a CNN with different spatial and temporal features that computes features based on changes from the peak reaction was proposed in [31]. Using generic feature extraction such as SIFT before passing it to a neural network can give decent results, with an 80% accuracy on BU-3DFE [32]. Deep convolutional neural networks can lead to a high accuracy when validated on the same dataset [33].

The use of preprocessing image techniques, such as image segmentation and histogram equalization, can increase the accuracy by a few percentage points. Detection accuracy can be improved by concatenating several features, or several neural networks. The authors of [26] used autoencoders in the first two layers to concatenate the texture and geometric features in order to improve accuracy. In [34], the authors merged VGG with *ResNet 50* to produce an output slightly better than both. Data augmentation has also been shown to improve the detection rates [42]–[44]. For example, [44] improved the accuracy by using rotations, translations, and other transformations on the original images to augment the size of the datasets.

The residual network, proposed in [35], is one of the most popular networks for computer vision. Their network can avoid the vanishing gradient problem with the deep network. *ResNet 34* has been used in multiple works to predict facial expressions, such as in [36] and [37]. Other types of residual networks are also used to detect facial expressions, such as in [38], where a generative adversarial network is used to generate features based on input images combined with *ResNet 50* for classification, and [39], which uses a region attention network to extract features that are sent to a *ResNet 16* classifier. Most of the techniques based on *ResNet* result in marginally better results on average, with a trade-off in larger preprocessing steps. References [40] and [41] used the VGG neural network to detect facial expressions with a high accuracy, but VGG tends to be enormous and requires a large amount of memory, which is not optimal for performance- and memory-limited low-power devices.

Another approach to detecting facial expressions using deep learning is to create a custom neural network. Building a custom neural network can be beneficial since it allows us to tweak it to our intended use case, whether for speed, accuracy, or temporal stability. CNNs can be used to detect spatial features with a good accuracy, as seen in [45], [47]–[49], while smaller networks can trade in some accuracy to reduce the processing time [50], [51]. Another approach is to use a long short-term memory (LSTM) neural network to detect temporal features, such as in [46]. However, this requires training and testing on a sequence of images instead of one image at a time, and is more computationally expensive than a similarly sized neural network.

E. OTHER FEATURES

Other features include advanced facial expression detectors that use specialized sensors, such as 3D and thermal sensors combined with normal cameras [53]. However, the reliance on specialized hardware to augment the capability of an RGB camera sensor makes the solution more expensive and less universal than a simple camera.

III. THE REFER ALGORITHM

Unlike most of the previously mentioned approaches, the two proposed techniques work in real time and are capable of classifying emotions on-the-fly, not just in controlled environments, such as the popular CK+ dataset. To highlight the advantages of the proposed approaches, we ran them on our own dataset [54] (see Section V-A for details) and on three publicly available datasets: CK+, MUG, and KDEF [55]. Another drawback of the other FER techniques is that they are usually optimized to work on individual datasets only and do not generalize well to unfamiliar data sources [42]. The two proposed approaches obtain significantly better results than the state-of-the-art in that aspect.

REFER takes the geometric approach in facial expression recognition to interpret human emotion. It works in real time by using highly optimized computer vision libraries for face landmark detection and our own simple but powerful descriptors using logarithmic, distance, and orientation scaling. This approach allows our algorithm to work with virtually any resolution under different poses at variable distances from the camera. Our approach is fast enough to work in real time, even on low-cost mobile devices. Running in real time is very important for the type of application that requires instantaneous facial expression detection.

The typical automatic facial expression detection algorithm using geometric features is shown in Figure 1. The two most popular approaches for face detection are Viola-Jones using Haar cascades [56] and local binary pattern (LBP) cascades, with Viola-Jones offering a higher accuracy and LBP requiring less computational power. Then, the key face points are extracted from the detected face using ASM. After the key point positions are extracted, the face descriptor is calculated based on the distances between different key points. After this point, the descriptors are either used to train the model if the facial expression is known, or to infer if the model is already trained. To train the model, we can either directly use the expression or use the facial action coding system (FACS), which codes the movement of individual features in the face [57], [58].

References [59] and [60] used a similar structure to construct their facial expression recognition system. The reported maximum accuracy in [59] was 85%, while the reported maximum accuracy in [60] was 80.9%. Reference [59]

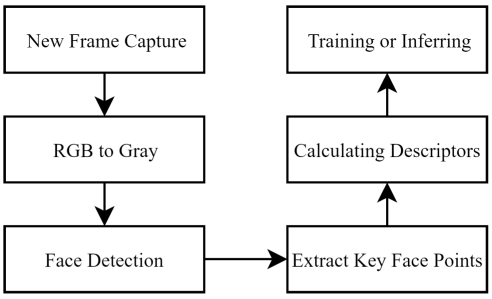


FIGURE 1. Typical workflow for automatic facial expression recognition algorithm using geometric features. Initially, each frame is obtained from stored pictures or videos; then, the image is converted from RGB to gray. The face is then detected, and feature points are extracted. From these data, the descriptor is typically calculated based on the distance or angles. Finally, the descriptors are used either for training or to infer the facial expression.

Algorithm 1 REFER Processing for One Video Frame

Input: Red, green, blue pixel matrices R, G, B
Output: P_1,P_2,...P_7

$K \leftarrow 0.299 \times R + 0.587 \times G + 0.114 \times B$ (1)
$[F, (x_c, y_c)] \leftarrow \text{VJH}(K)$ (2)
L = width(F) (3)
$\{[x_0, y_0], [x_1, y_1], \ldots, [x_{67}, y_{67}]\} \leftarrow \text{ASM}(F)$ (4)
$H = \frac{(x_0 - x_c)^2}{(x_{16} - x_c)^2}$, (5)
$V = \frac{(y_{27} - y_{28})^2}{(y_{29} - y_{28})^2}$, (6)
foreach *local region* $\mathcal{R}_i$ **do** // See Table 1
  $N = |\mathcal{R}_i|$ (8)
  $\tilde{x}_c \leftarrow \sum_{n=0}^{N} \frac{x_n}{N}, \tilde{y}_c \leftarrow \sum_{n=0}^{N} \frac{y_n}{N}$ (9)
  for $n \in \mathcal{R}_i$ **do**
    $d_n^2 \leftarrow \frac{(\tilde{x}_c - x_n)^2 + (\tilde{y}_c - y_n)^2}{L^2}$ (11)
    if $x_n \leq \tilde{x}_c$ **then**
      $\tilde{d}_n^2 \leftarrow d_n^2 \times H^2$ (13)
    if $y_n \leq \tilde{y}_c$ **then**
      $\tilde{d}_n^2 \leftarrow \tilde{d}_n^2 \times V^2$ (15)
    $s_n \leftarrow \log \tilde{d}_n^2$ (16)
  $\boldsymbol{p}_i \leftarrow \text{pSVM}(\{s_n : n \in \mathcal{R}_i\})$ (17)
$V_p = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_8]$ (18)
$\boldsymbol{P} \leftarrow \text{pSVM}(V_p)$ (19)

reported a maximum speed of 2.4 FPS on mobile hardware. Reference [61] used an active appearance model with head pose estimation and normalization to detect faces at angles. REFER produces a higher accuracy rate in both cross-validation tests and is significantly faster.

A. OVERVIEW OF REFER

The overall REFER algorithm requires an input colored image K and outputs the detected facial expression. The REFER algorithm is described in Algorithm 1. From image K, we perform a few preprocessing steps described in III-B. We then compute the local descriptors as shown in III-C. After obtaining the local descriptors, we perform the first round of local facial expression detection, from which we compute the global descriptors and global output of the algorithm, as seen in III-D. The result of the algorithm can be adjusted based on the previous frame to provide stability in the results in Section III-E. Finally, in III-F we show how using AVFD works to provide significant performance gains on computationally limited devices.

TABLE 1. The local region's ASM points.

Feature	Key-Point Number
All points	$\mathcal{R}_0 = \{i : 0 \leq i \leq 67\}$
Jawline points	$\mathcal{R}_1 = \{i : 0 \leq i \leq 16\}$
Left eyebrow points	$\mathcal{R}_2 = \{i : 17 \leq i \leq 21\}$
Right eyebrow points	$\mathcal{R}_3 = \{i : 22 \leq i \leq 26\}$
Left eye points	$\mathcal{R}_4 = \{i : 36 \leq i \leq 41\}$
Right eye points	$\mathcal{R}_5 = \{i : 42 \leq i \leq 47\}$
Outer mouth points	$\mathcal{R}_6 = \{i : 48 \leq i \leq 60\} \cup \{64\}$
Inner mouth points	$\mathcal{R}_7 = \{61, 62, 63\} \cup \{65, 66, 67\}$

B. PRE-PROCESSING

To train or detect faces using REFER, we first have to obtain the image frame, either from still images or from a live camera feed. The image is then converted from RGB to gray using the popular equation given on line 1, where K is the resulting gray image, and R, G, B are the red, green, and blue pixel matrices, respectively. We detect the face region using Viola-Jones with Haar cascade features, denoted by $[F, (x_c, y_c)] = \text{VJH}(K)$, which takes an image matrix K as the input and returns a new image matrix F containing only the face area, as well as the position (x_c, y_c) of the center of the face within the overall image.

C. LOCAL DESCRIPTORS

Instead of using all ASM points as one set, we used these points to create several subsets. Figure 2 shows how the descriptors are extracted. The big green box is the face detected using Viola-Jones. The yellow dots with numbers on top are the face points detected using ASM with `dlib`. The smaller blue boxes are manually added to show what points each descriptor uses. We use 8 descriptors: one uses all the key points, while the others are partial descriptors that cover only one facial feature each. After calculating the distance between the key points, we scale the distance based on the detected face size and the perceived orientation of the face.

Unlike most approaches that use only one classifier, REFER uses a two-step ensemble of classifiers to detect facial expressions with a high accuracy under different head poses. The pseudo-code is shown in Algorithm 1. The variables will be discussed in the following subsections.

ASM is used to detect and track 68 key face points. From these 68 key points, we generate a global descriptor that uses all 68 key points, and we also generate local descriptors for each part of the face. The key points used for each descriptor are given in Table 1.

The proposed descriptors are designed to be robust to changes in pose and scale. This allows the facial expression

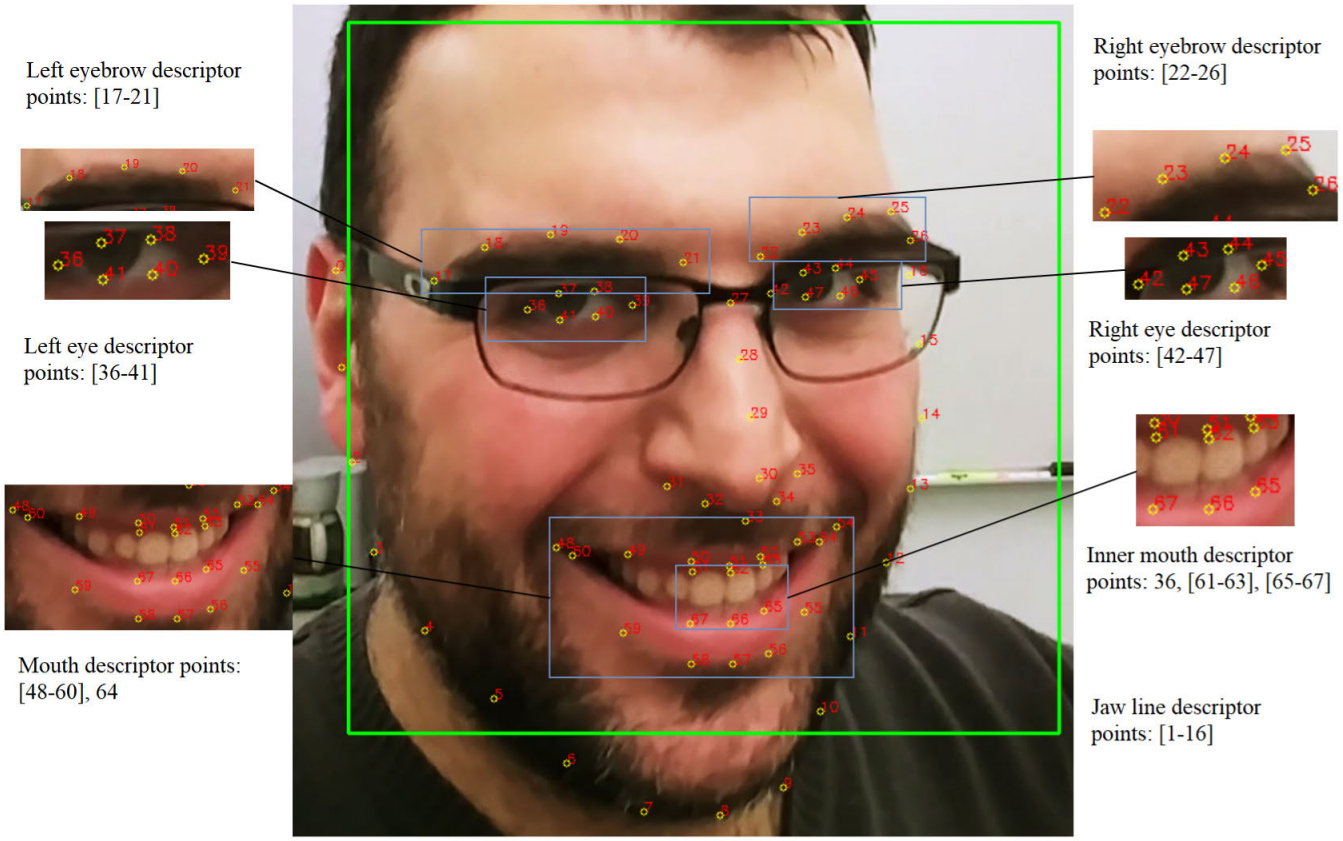


FIGURE 2. Volunteer descriptor example: the green square is the detected face, the yellow points are the detected key points, [these two are automatically drawn in real-time], and the blue rectangles are manually added to visualize the different face descriptors.

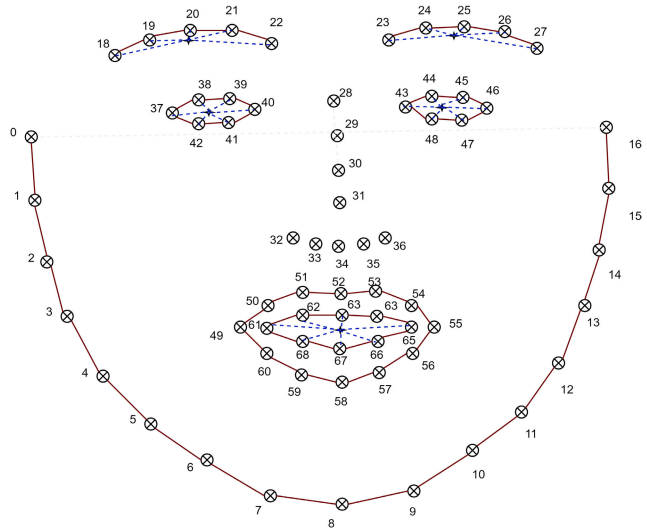


FIGURE 3. Location of ASM points on the face and the local descriptor subsets. The red lines connect the ASM points that are part of the same set, while the blue dotted lines illustrate the distance measures used in our descriptors. Stars represent the mean points of each descriptor. Note that some subsets and mean points are not shown to improve clarity.

to work better in the wild, where a camera could be pointed at almost any angle and distance. To achieve this, we scale the distance for both size and rotation, and finally, we apply a logarithmic kernel. The scaling is performed with respect to a virtual mean point $[\tilde{x}_c, \tilde{y}_c]$ for each local descriptor, which is obtained by averaging the location of the points in the descriptor (line 9). Figure 3 shows some of the sets and the location of the mean points for these sets. We only apply vertical scaling to the point above the calculated center and horizontal scaling to the point to the left of the calculated center. We compute the Euclidean distance of each key-point, as shown on line 11. This distance is normalized by the size of the face region, which helps provide invariance to the image resolution and to the distance between the subject and the camera.

To compensate for the horizontal and vertical orientations of the head pose, we multiply the scaled descriptors by an approximate ratio of the rotation. Horizontal scaling corresponds to lines 5 and 13 of Algorithm 1, where H is the horizontal scaling factor and $\tilde{d}$ is the scaled distance. Point 0 from the detected key points is the point at the top left of the face, and point 16 is the point at the top right of the face. We compute the ratio between these distances and use them to scale the left part of the face. These two points were chosen because they do not correspond with any moving muscle, meaning that they will not move with respect to facial expressions, making them good reference points for scaling. When $H > 1$, the left side of the face is closer to the camera than the right side, and when $H < 1$, the right side is closer than the left. Scaling is then applied to the left side

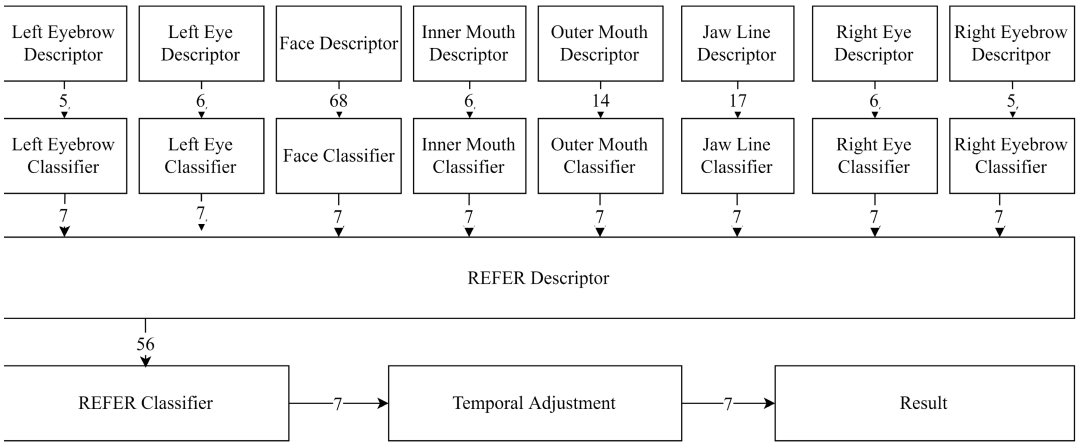


FIGURE 4. Two-stage classifier hierarchy: the arrows show the flow direction of classification, and the numbers represent the vector size for each step in the classifier. In the first stage, the classifiers are responsible for local-level predictions, while the second stage is responsible for global classifications.

to realign the distances in the face to be similar to the frontal view.

We then compute the ratio of the vertical scaling, which compensates for the face tilting back and forth. The vertical rotation ratio V computed on line 6 is based on the vertical positions of points 27, 28, and 29. These points are chosen since they are nose points that do not move with facial expressions, but they do move only when a face tilt is detected. These points are also equidistant in the full frontal view. The scaled distances are then computed using the equation on line 15. When $V > 1$, it means that the face is tilted forward or looking down on the camera. If $V < 1$, the face is tilted backward or looking up to the camera. We apply vertical scaling to the scaled distances for all key points above the central pixel to compensate for the tilt.

Finally, since the descriptors are based on relative distance, all the values are between 0 and 1. To increase the separation between the descriptors, we take the logarithm of $\tilde{d}$, as shown on line 16.

D. ENSEMBLE SVM CLASSIFIER AND TEMPORAL ADJUSTMENT

After obtaining the multi-descriptors from each local facial feature, we train each classifier with a *pSVM* classifier [62]; hence, every local feature will generate its probability prediction as an output probability vector. These probabilities are then used as training for the second-stage classifier, which takes all probabilities from all facial features as an input and builds a decision hyperplane based on these probabilities. Figure 4 shows the overall hierarchy of the descriptors and classifiers. The first stage of classifiers uses several parallel independent local descriptors, each with their key points. After obtaining the first descriptors, we predict the probability of the facial expression based on each classifier. This gives us an output probability vector of each facial expression for each local facial feature. These output probabilities are then used as input for the second-stage global classifier, which produces a global probability based on the local features. This will give a better prediction than the local classifiers. The output of the second-stage classifier is also represented in probability.

Algorithm 2 Temporal Adjustment of the Classification

Input: $\boldsymbol{P}$
Output: $\boldsymbol{P'}$

$$\boldsymbol{P'} \leftarrow \boldsymbol{P} \quad (1)$$
$$P'_{e^{(t-1)}} \leftarrow P_{e^{(t-1)}} + a \quad (2)$$
$$P'_{e^{(t-2)}} \leftarrow P_{e^{(t-2)}} + a \quad (3)$$
$$\boldsymbol{e}^{(t-2)} \leftarrow \boldsymbol{e}^{(t-1)} \quad (4)$$
$$\boldsymbol{e}^{(t-1)} \leftarrow \text{argmax}_i(P'_i) \quad (5)$$

E. TEMPORAL ADJUSTMENT

Since facial expressions do not change quickly within a span of a few consecutive frames, we can assume that previous predictions would likely be similar to the current prediction. Therefore, to take advantage of this temporal property, we boost the current prediction probability vector with the result from previous predictions. This allows the result to be more stable. For example, when deciding between disgust and happy, both cases can include a partially open mouth, especially if the smile is not complete or transitioning from a different expression. In this case, we noticed that the prediction jumps back and forth between these two expressions, and adjusting the prediction probability based on previous predictions stabilizes the classification.

Algorithm 2 shows the steps of the temporal adjustment. It takes the probability vector $\boldsymbol{P}$ generated by Algorithm 1 as the input and outputs a corrected decision vector $\boldsymbol{P'}$. Initially, the prediction indices $e^{(t-1)}$ and $e^{(t-2)}$ are set to 2, the index of the neutral facial expression. To avoid unnecessary computations, $\boldsymbol{P'}$ is not renormalized, and the final prediction is given by $\text{argmax}_i P'_i$. The constant a is selected empirically and set to $a = 0.4$.

F. ALTERNATE VIDEO FRAME DETECTION

In our tests, running this algorithm on a PC from a live video capture resulted in 30 FPS, which is the maximum rate. However, running this on a Raspberry Pi 4, the FPS dropped to 6.3. This is 2.6 times faster than the ASM-based paper [59], which reported an average of 2.4 FPS, although they were using a different SoC. The process of running feature extraction and classification is largely serial. The performance results are discussed in depth in Section V-E.

Most modern mobile devices offer multiple cores, even affordable ones such as the Raspberry Pi 4. To improve the performance on mobile devices, we can split the tasks between all 4 cores to maximize CPU utilization, increase performance, and reduce latency.

Normally, the process for facial expression recognition is a simple feed-forward technique, but due to the many steps required to complete the process, the execution time is relatively large, resulting in a low FPS. Using alternate video frame detector (AVFD), the work can be divided among all the available cores. One core is assigned to read the camera input, convert it from color to gray, and put the frame buffer into a shared last in first out (LIFO) queue. The core can also retrieve the decision from the shared queue, sort, and apply the temporal adjustment. The three other threads each pop the last frame and run the remaining steps to REFER on their own assigned frame. Since Raspberry Pi only contains 4 cores, we will use three threads to compute REFER, but if more cores are available, we can easily divide the work into more cores. This technique can hide some of the latency of real-time facial expression recognition, since the processes of reading frames from webcams or stored video no longer wait for the previous prediction to finish.

Python has a limitation in the multithreaded workload due to the Global Interpreter Lock (GIL). The GIL effectively locks the interpretation of the Python script into machine code to only one thread; hence, it allows only one thread to run a Python command concurrently, so a multithreaded Python code can only use 1 core effectively. To address this limitation, we used an external C compiled library. Not only do these libraries run faster than a regular python expression, but they also run outside the locks of the GIL, thus effectively allowing the use of multiple cores. In our tests, we gained a 2.22 times speed increase, with a perceivable latency reduction (time between facial expression is performed and the result appears on the screen). The CPU utilization for a single-threaded execution was 30% on average with a memory utilization of approximately 200 MB. When using AVFD, the CPU utilization jumped to 85%, with memory allocation closer to 550 MB. This is because we had to keep a copy for the face detector, dlib, and our classifiers for each thread in memory; otherwise, we would have race conditions when threads are competing to access the classifiers that cause crashes or slowdowns.

There was no need for parallelization on the laptop PC, since we were already able to run the algorithm at the maximum frame rate of the camera. Thus, there would be no benefit to parallelize the code on the PC.

Algorithm 3 FERNet Processing for One Video Frame

Input: Red, green, blue pixel matrices R, G, B
Output: $Q_1, Q_2, \ldots, Q_7$
$K \leftarrow 0.299 \times R + 0.587 \times G + 0.114 \times B$ (1)
$F \leftarrow \text{VJH}(K)$ (2)
$\mathcal{P} \leftarrow \text{ASM}(F)$ (3)
foreach *ASM point* $n_i \in \mathcal{P}$ **do**
  foreach *ASM point* $n_j \in \mathcal{P}$ **do**
    $u \leftarrow [x_{n_i}, y_{n_i}]$ (6)
    $v \leftarrow [x_{n_j}, y_{n_j}]$ (7)
    $C_{i,j} \leftarrow \frac{u.v}{\|u\| \times \|v\|}$ (8)
$Z \leftarrow \text{Imresize}(F, 68 \times 68)$ (9)
$B \leftarrow \text{LBP}(Z)$ (10)
$V_q = [C, B]$ (11)
$\boldsymbol{Q} \leftarrow \text{FERNet}(V_q)$ (12)

IV. THE FERNet ALGORITHM

As shown before, there have been many attempts to create a neural network dedicated to detecting facial expressions. The most popular approaches are *ResNet 34* and *ResNet 50*. These two networks are large and computationally expensive since they require approximately 20 and 23 million parameters, respectively. Thus, we propose FERNet, a small, computationally efficient and speedy neural network capable of detecting facial expressions from a wide range of angles with a relatively low number of computations with 325,479 parameters with a very minimal impact on accuracy.

The steps for FERNet are shown in Algorithm 3 and are described in the following subsections. REFER and FERNet share the first 7 steps in the algorithm.

A. PREPROCESSING AND FACE DETECTION

To perform automatic facial expression recognition, we have to initialize the camera, capture a new image, and transform it from a color image to a gray image (line 1). We then detect the face region in the image using the Viola-Jones algorithm.

B. GEOMETRIC FEATURES EXTRACTION

To obtain the most important geometric features, we use ASM to detect 68 pertinent points around the face. We then compute the cosine similarity for every pair of points using Line 8 in Algorithm 3, where u and v are any points within the 68 detected using ASM, and $C_{i,j}$ is the cosine similarity between two ASM points n_i and n_j. Since ASM gives 68 points, the resulting cosine similarity is a 68×68 matrix representing the cosine similarity from every ASM point to every ASM point. The benefits of using cosine similarity are that it is independent of the image scale and it does not exhibit a large variation in different poses.

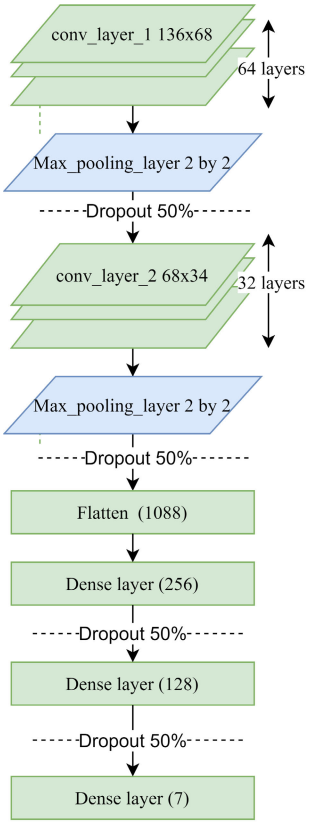


FIGURE 5. FERNet composition: the three-dimensional rectangles represent the convolutional layers, while the flat rectangle represents the dense flat layers.

C. TEXTURE FEATURE EXTRACTION

Texture features can be good at detecting facial expressions, as they can detect ridges and wrinkles on the face, making them a good fit for facial expression recognition. However, analyzing texture features can be computationally expensive, such as when using a Gabor filter; hence, this work uses LBP for texture feature extraction, as it is effective, computationally inexpensive, and works in non-uniform lighting [13]. The use of LBP is denoted as LBP() in Algorithm 3. A size 3 ring is used for LBP, as we found it to be a good balance between performance and accuracy, meaning we only look at one pixel and the directly adjacent 8 pixels to obtain the LBP image. To compute the LBP on each pixel, we first resize the face image F to 68×68 using the function Imresize with bilinear interpolation, which results in a scaled face image Z. We then begin computing the LBP of the scaled face image Z by comparing the value of each pixel with its neighboring pixels. Based on which neighboring pixels are larger than the central pixel, we obtain a value between 0 and 255. The resulting values indicate the direction of intensity. *LBP* is explained in detail in [13].

D. FERNet

After obtaining both the texture and geometric features, we merge both features. Both features are represented by a 68×68 matrix, resulting in a concatenated matrix V_q of size 136×68. Matrix V_q is then given as input to the neural network FERNet shown in Figure 5. The FERNet architecture begins with a convolutional layer with rectified linear activation and 64 feature maps. We then implement a 2-by-2 max-pooling layer to reduce the size of the data to a fourth, followed by a dropout layer with a dropout parameter of 50%. As shown in Fig. 5, we repeat the same structure another time to further reduce the data, but this time with 32 feature maps. After that, we flatten the convolutional network to a dense layer of size 1088. We then add another dense layer of 256 neurons with a dropout of 50% and then another dense layer of 128 neurons with a dropout of 50%.

Finally, the output layer has 7 outputs, each providing the probability Q_i that the facial expression i was observed in the input.

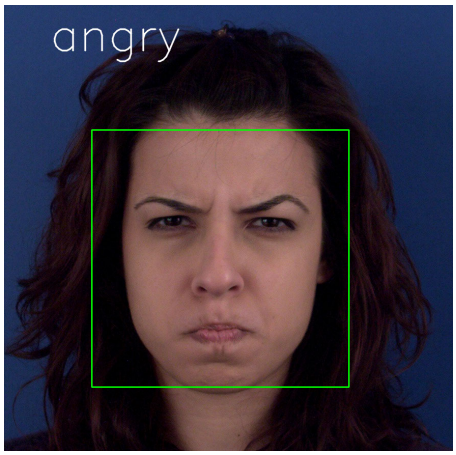


FIGURE 6. Example of correct classification from the MUG dataset with angry faces trained with CK+ datasets.

FIGURE 7. Example of correct classification from CK+ with fear faces. [17].

V. RESULTS

To verify the algorithms, we implemented the training code using Python on a Windows 10 laptop PC equipped with an i7-8750h Intel CPU and 32 GB of RAM. We used multiple libraries including `OpenCV` for frame capture and face detection, `dlib` to obtain ASM points [63], and `scikit-learn` to train the two-stage SVM model and test the results [64]. Tensorflow is used to train and test FERNet. For accuracy and performance testing, we used the same previously mentioned laptop PC, and we also tested the inference accuracy and performance using a Raspberri Pi 4 device with 4 GB of RAM.

A. TRAINING AND TESTING DATASETS

In our training and testing, we used four different datasets, all publicly available. We used three existing datasets: CK+, MUG, and KDEF. We also created a custom dataset called angled posed facial expression (APFE) [54]. This dataset contains approximately 15,000 frames of video filmed under different angles at peak expressions for 4 males with no acting experience aged between 25 and 30. The videos were shot at a resolution of 640 × 480 and at 30 FPS using a `Logitech c270` camera. The camera was fixed on top of a screen, and the volunteers were told to look at the camera and move their faces in concentric circles while performing facial expressions with different severities, to look up and down, and finally to make sure that the camera captured their faces from several angles. We also captured a 1080p video from an LG G6 smartphone to use for testing only in order to verify that the proposed algorithms work on different resolutions and different aspect ratios.

The CK+ dataset [17] contains 123 different subjects, both males and females, shot directly with a resolution of 640 × 490, for a total of 10,000 image frames. Each shot starts at neutral and increases until peak expression. The MUG dataset [52] contains images and videos for 52 different males and females shot at an 896 × 896 resolution at 20 FPS with 1032 videos and more than 100,000 images. Each video begins and ends with a neutral expression and peaks in the middle. The KDEF dataset contains 4900 photos of 70 subjects with 7 different facial expressions, each viewed from 5 different angles.

It is challenging to use *transfer learning* to train networks for facial expression recognition since the networks tend to overfit to the original data, making it difficult to target a new dataset [37]. Hence, in all our tests, we trained FERNet and *ResNet* from scratch.

B. DATA AUGMENTATION

Neural networks are well known to require a large amount of data to fine tune. Data augmentation methods that increase the size of the dataset can lead to an improved accuracy, as seen in [43], [44], and [46]. To see the effect of data augmentation on the detection accuracy, we implemented 5 types of augmentation for every image in each of the datasets. The 5 augmentations are image flip, random brightness adjustment, simultaneous flip and brightness adjustment, random positive rotation, and random negative rotation. In the image flip, we simply flip the image from left to right. In the random brightness adjustment, we convert the image from RGB to hue-saturation-value (HSV) and multiply the saturation and value components of each pixel by a random factor between 0.5 and 3 and convert it back to RGB. Finally, in the image rotation augmentation, we randomly rotate the image by a random angle between 0° and 30° for positive rotations or between −30° and 0° for negative rotations. The augmentation results in a dataset that is 6 times larger than the original dataset. Note that for all upcoming graphs and tables, the accuracy results are when using the data augmentations mentioned above, except for Table 2 where we only used the original dataset with no augmentation.

TABLE 2. Accuracy percentage when training from original images and testing with randomly rotated images.

Angle range	CKP	MUG	KDEF	APFE
0°	75.74%	96.69%	73.44%	97.37%
[−15°,15°]	72.74%	95.84%	69.52%	94.81%
[−30°,30°]	71.07%	93.96%	66.55%	93.29%
[−45°,45°]	69.15%	91.11%	65.14%	93.17%
[−60°,60°]	68.68%	87.78%	61.8%	91.88%
[−75°,75°]	67.48%	85.25%	61.48%	92.19%

C. ACCURACY OF REFER

To test the accuracy of the proposed REFER algorithm, we trained REFER under different conditions to test for cross-validation accuracy and robustness for rotation.

First we tested for cross-validation accuracy by training with 90% of the data for every dataset independently and validating on 10% of the remaining data. We repeated each test three times and took the average of these runs. The results are shown in Table 3. While using REFER on the CK+ dataset, we obtained an accuracy of 80% in cross-validation when using data augmentation. Similarly, we obtained accuracies of 97.7%, 74.3%, and 97.7% for the APFE, KDEF, and MUG datasets, respectively, with the same 90-10 split between training and validation. The accuracy of predictions from the videos was measured frame by frame, meaning that we produce a classification for every frame and compare it to the actual facial expression.

Second, we tested the robustness to rotations by training each dataset with non-rotated images and testing on randomly rotated images. The results in Table 2 show the accuracy of REFER on all datasets for different ranges of uniformly distributed random rotations, which indicate that REFER is robust to rotations in wide ranges of angles.

These results show that this algorithm works well with the different camera systems, resolutions, apertures, distances, aspect ratios, and lighting, which are covered by the four considered datasets, and that it can also make accurate predictions at an angle. Figures 6, 7, 8a, 8b and 8c show some correctly predicted samples from the MUG, CK+, and APFE datasets.

Table 3 provides a comparison of the state-of-the-art literature for different approaches. Both REFER and FERNet exhibit some of the highest accuracies in cross-validation testing compared to other geometric- or neural-network-based approaches while having the fastest detection rate.

Table 4 shows the internal probability from classifiers in action when attempting to predict the facial expression from Figure 8c. We can see in Table 4 that individual features gave a different prediction without a high confidence. While the second stage classifier not only chose the correct facial expression, it also did so with a high confidence. REFER

TABLE 3. Comparing REFER and FERNet with the state-of-the-art.

Dataset	Method	Accuracy	Real-time	Feature types	Classifier (# of param)	Validation
CK+	ASM [59]	77.5%	2.4 FPS (APQ8064A)	geo	SVM	10-fold
	ASM+2D-DCT [60]	77%	no	geo+texture	SVM	5-fold
	GA-SVM [10]	88%	N/A	geo	SVM	cross-validation
	CNN [44]	97%	no	CNN	CNN (9.4M)	cross-validation
	GAN+CNN [38]	99%	N/A	GAN + CNN	CNN	cross-validation
	HoG+PCA [49]	82%	no	geo	DNN	10-fold
	Resnet 34	43%	no	CNN	CNN (21M)	cross-validation
	Resnet 50	89%	no	CNN	CNN (23.6M)	cross-validation
	REFER	**80%**	**14 FPS (RPi4)***	**geo**	**SVM**	**cross-validation**
	FERNet	**83%**	**14 FPS (RPi4)***	**geo + CNN**	**CNN (0.3M)**	**cross-validation**
MUG	VGG-face [45]	94.5%	no	VJ+VGG-face	VGG-face (138M)	10-fold
	GA-SVM [10]	88%	N/A	geo	SVM	cross-validation
	CNN [50]	87%	N/A	CNN	CNN	cross-validation
	HiNet [51]	88.6%	N/A	CNN	CNN (1M)	cross-validation
	DWMV [21]	96%	N/A	VJ+SURF	Ensemble	cross-validation
	Resnet 34	55%	no	CNN	CNN (21M)	cross-validation
	Resnet 50	98%	no	CNN	CNN (23.6M)	cross-validation
	REFER	**97.7%**	**14 FPS (RPi4)***	**geo**	**SVM**	**cross-validation**
	FERNet	**98%**	**14 FPS (RPi4)***	**geo + CNN**	**CNN (0.3M)**	**cross-validation**
KDEF	CNN [44]	75.85%	no	CNN	CNN (9.4M)	cross-validation
	VGG. [41]	72%	no	VGG	VGG (138M)	cross-validation
	Resnet 34	28%	no	CNN	CNN (21M)	cross-validation
	Resnet 50	66%	no	CNN	CNN (23.6M)	cross-validation
	REFER	**74.3%**	**14 FPS (RPi4)***	**geo**	**SVM**	**cross-validation**
	FERNet	**76%**	**14 FPS (RPi4)***	**geo + CNN**	**CNN (0.3M)**	**cross-validation**
APFE	Resnet 34	47%	no	CNN	CNN (21M)	cross-validation
	Resnet 50	97%	no	CNN	CNN (23.6M)	cross-validation
	REFER	**97.7%**	**14 FPS (RPi4)***	**geo**	**SVM**	**cross-validation**
	FERNet	**99%**	**14 FPS (RPi4)***	**geo + CNN**	**CNN (0.3M)**	**cross-validation**

* Frames-per-second performance was measured on live video capture.

TABLE 4. Breakdown of the classification probabilities for each descriptor for the example of Fig. 8c.

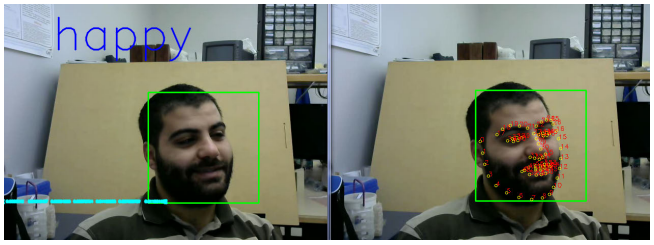
Descriptors	Angry	Neutral	Disgust	Fear	Happy	Sad	Surprised
Whole face (ASM)	0.042	0.004	0.23	0.007	0.69	0.001	0.0107
Jaw line	0.137	0.196	0.161	0.172	0.029	0.206	0.095
Inner mouth	0.021	0.033	0.279	0.095	0.388	0.023	0.158
Outer mouth	0.015	0.036	0.406	0.308	0.007	0.0344	0.19
Left eye	0.057	0.056	0.223	0.062	0.425	0.163	0.011
Left eyebrow	0.279	0.104	0.27	0.054	0.103	0.172	0.015
Right eye	0.03	0.024	0.251	0.0429	0.585	0.057	0.006
Right eyebrow	0.12	0.215	0.141	0.122	0.133	0.228	0.037
REFER	0.0057	0.0026	0.975	0.004	0.004	0.002	0.003

correctly predicted that ''disgust'' is the correct expression, as it has built a decision hyperplane that deduces which descriptors are more relevant for each expression.

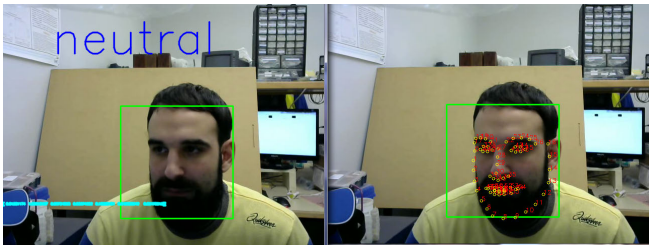
For cross-dataset validation, we trained REFER on one dataset and tested the accuracy on the remaining datasets. The results are shown in comparison with FERNet and *ResNet 34* in Figure 11. Note that for graph readability, we omitted the results with KDEF since it showed a similar trend. We also omitted the results of testing without data augmentation for graph readability. For example, training with CK+ and testing the accuracy on the MUG dataset, we obtained an accuracy of 70.15%, which increased to 72.1% when we used data augmentation. In all cases, we observed a small improvement when using data augmentation, which does not, however, have a bearing on the ranking of the solutions. We also used previously non-trained video sequences shot at 1920 × 1080 from the front-facing camera of LG G6 at different distances and locations while moving the camera around the face to see all angles while ensuring that all the faces were captured within the shot. We copied these shots to a PC and tested them with REFER, obtaining an accuracy of 97%.

While REFER achieves a good accuracy, there are several cases of failure in REFER that can be improved in the future. Figure V-C shows an instance where a face was not detected due to the high tilt angle. The Viola-Jones failed to detect a face failing the remainder of the algorithm. Figure 8d shows a misprediction, where REFER detected the face as a sad face while it is a surprised face. We can notice that the failure was in part due to a failure in ASM to accurately register points when at a large angle, the black beard and relatively dim lighting might have also negatively affected the accuracy of ASM and subsequently the remaining steps in the classification.

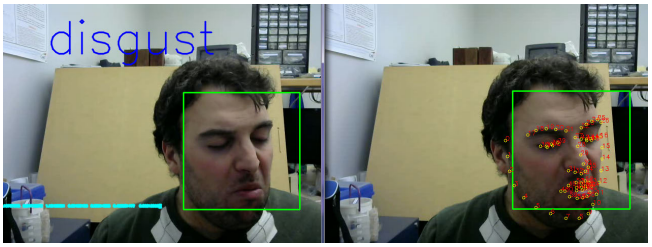
There are few instances where the detection fails, even with proper face registration due to the difference in how people's face moves while making facial expressions.



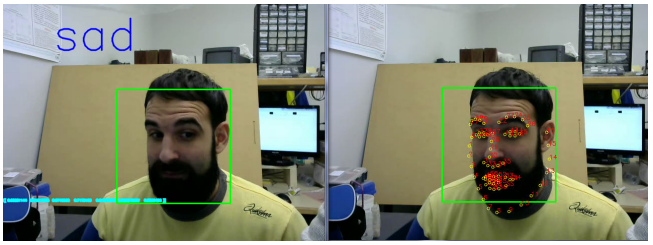
(a) Volunteer happy example correctly classified.



(b) Volunteer neutral example correctly classified



(c) Volunteer disgust example correctly classified.



(d) Mispredicted surprise with sad face due to a failure in registration of the ASM algorithm.

FIGURE 8. Example of REFER in action.

FIGURE 9. Failed to detect faces due to the large pose angles, no prediction can be made if faces are not detected.

D. ACCURACY OF FERNet

Since FERNet is based on a CNN, we compared it with two popular approaches: *ResNet 34* and *ResNet 50*. We implemented all three networks using TensorFlow with the same dataset as REFER. We first detected the face in the image using Viola-Jones face detection. Then we resized the area of the detected face to fit the input layer of the network. Thus, we gave each network its ideal data shape to work with.

Figure 10 shows the epoch accuracy of FERNet compared to *ResNet 34* and to REFER. In these cases, the same data is used for both training and testing. Since REFER is an SVM-based approach, there is no need for epochs; hence, the value is constant. FERNet gained its accuracy much faster than *ResNet 34* with epochs and required a significantly lower number of epochs to settle. While *ResNet 34* achieved close to a 100% accuracy on the training set in all datasets, it performed poorly in the cross-validation tests, which indicates overfitting. FERNet training and cross validation accuracy were very close, indicating very little to no overfitting.

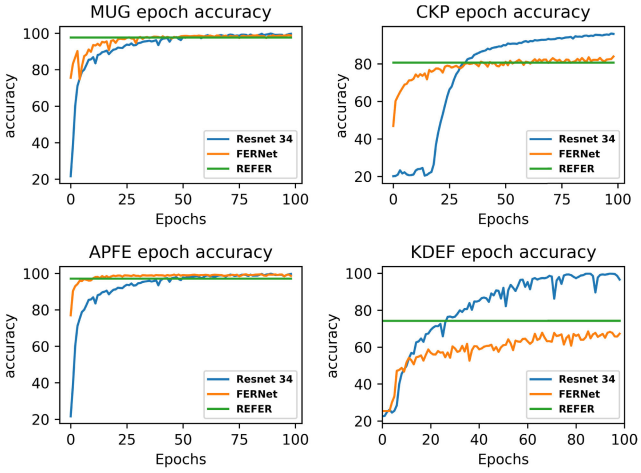


FIGURE 10. Accuracy for the first 100 epochs comparison between *ResNet 34*, FERNet, and REFER for each dataset.

To compare the cross-validation accuracy, we used 90% of the data for training and 10% for validation with 499 epochs for all cases, except for REFER, where we used the same 90-10 split for training and testing but without having to use epochs, since it is based on SVM. The results are the bottom four of Table 3 for each dataset. This shows that FERNet outperforms *ResNet 34* in all tests and *ResNet 50* in three out of four datasets in cross-validation testing.

Figure 11 shows the results of cross-dataset validation accuracy. In these tests, we trained the network with one dataset and used the other two for testing. REFER has a significant lead in cross-dataset validation over other approaches, followed by FERNet. Both of our approaches performed significantly better than *ResNet 34* in all tests, with a better absolute accuracy of 20% or more.

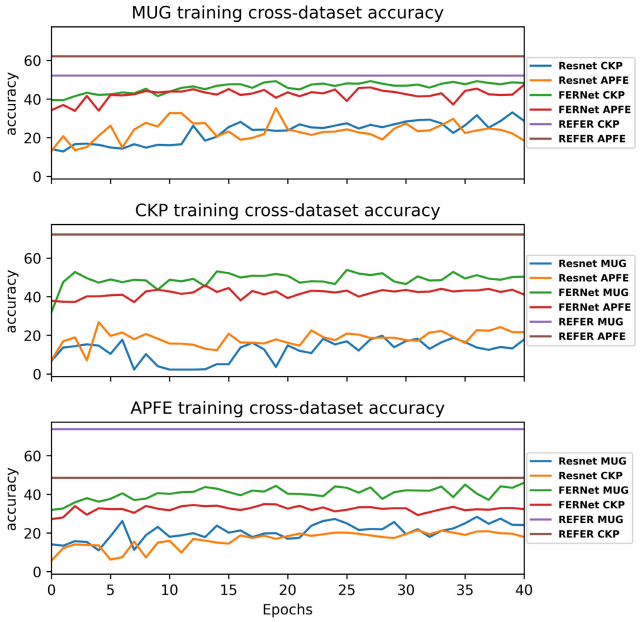


FIGURE 11. **Cross-dataset validation comparison between *ResNet 34*, FERNet, and REFER as a function of the training epoch. The top is when training using MUG and validating using the CKP and APFE datasets, the middle is when training using CKP and validating using the MUG and APFE datasets, the bottom is when training using the APFE dataset and validating using MUG and CKP. The accuracy of REFER does not vary since there is no notion of epoch in SVM training.**

ResNet 34 cross-dataset validation accuracy shows that there were no gains with the epochs; on the other hand, FERNet did show improvement with the cross-dataset training for every dataset. The combination of our smaller network with the large dropout rate ensures a better generalization outcome, and our simple descriptors give it a better starting accuracy, even at epoch 0. However, there is a large difference between the epoch accuracy and cross-dataset validation accuracy, indicating that current datasets do not have sufficient variations to allow for a generalized solution. Moreover, as seen in Figure 11, REFER performs better than the CNN-based approaches (FERNet and *ResNet 34*) in the cross-dataset validation. Therefore, it can be expected to perform better in less-controlled environments.

E. PERFORMANCE TESTING

One of the main benefits of using a smaller network is to have a network with a good real-time performance. Given that FERNet has 325,479 tunable parameters compared to 20 million parameters in *ResNet 34*, we expect the network to train faster and to run inference faster. To measure the performance, we used two systems. One laptop PC with an i7-8750h with 32 GB of RAM with an RTX 2060 GPU and a Raspberry Pi 4 with 4 GB of RAM. Training the network took 30 minutes for FERNet and 4 hours for *ResNet 34* when using the GPU to train the networks. Once those were trained, we ran the same network using a laptop. To run the networks on a Raspberry Pi 4, we saved the trained network to a file on a PC and then moved the file to the Raspberry Pi 4 and

TABLE 5. **Number of frames per second for ResNet 34, FERNet and REFER.**

Network	ResNet 34	FERNet	REFER
Laptop	15	30	30
Raspberry Pi 4	2.75	11.5-14	6.3-14

installed the required library, frameworks, and necessary system environment variables.

To measure the FPS, we ran a code that includes all the steps of automatic facial detection from start to finish, including opening the camera, obtaining a new frame, converting a color image to grayscale, detecting the face, running the processing and feature extraction, running the network, and finally obtaining the prediction. In the case of *ResNet 34*, we only need to detect the face and resize it to fit the input layer of the network and skip the remaining preprocessing steps. We measure the performance by counting the number of predictions divided by the number of seconds in the time elapsed. We ran the test for approximately 5 minutes of live video capture. Table 5 shows the performance summary when running the full stack of automatic facial expressions. On the laptop GPU, we found that running *ResNet 34* resulted in an average of 15 predictions per second, while the predictions when running FERNet are a stable 30 FPS, which is the maximum frame rate that can be achieved with the camera. On the Raspberry Pi 4 we ran three tests. Using *ResNet 34*, we determined that the average frame rate is 2.75 FPS. Running FERNet resulted in 11.5 FPS on the Raspberry Pi 4. Flattening the network and running it with TensorFlow-lite on the Raspberry Pi 4 resulted in 14 FPS.

The results show that, despite having more preprocessing steps and feature extraction than *ResNet 34*, FERNet ran up to 5 times faster with a very small detriment to the accuracy, even surpassing both *ResNet* networks, with the cross-validation accuracy indicating better results on the fresh data. Running REFER on the Raspberry Pi 4 results in 6.3 frames per second when using a single-threaded approach. Using AVFD on Raspberry Pi 4 resulted in 14 frames per second. Note that while AVFD does improve the throughput, the latency of detection will not improve since the detection pipeline still needs to go through all the steps before reaching a result, but it can reduce the latency of image capture since we do not have to wait for the current frame to finish to begin working on the next frame. Another thing to note is that both REFER while using AVFD and *ResNet 34* require more than 80% of the CPU time when running. Single-threaded REFER and FERNet require approximately 30% and 40% the CPU time, respectively.

From the performance and accuracy results, we can see that FERNet is the fastest approach with the lowest latency for detecting facial expressions with a high accuracy on familiar data, making it ideal for applications where we can tune the network before applying, while still working well on new data. REFER truly works well with fresh data, as shown in the cross-dataset validation, while being fast enough for many applications, while less accurate than FERNet or ResNet on familiar data.

VI. CONCLUSION

This paper presented two novel approaches for real-time automatic facial expression recognition called REFER and FERNet. The first accurately predicts facial expressions at different angles and distances with up to a 97% accuracy in real time on the APFE dataset, even on mobile hardware, such as a Raspberry Pi 4.

REFER contains four improvements that can be used independently: custom descriptors based on geometric features that allow detection at different angles and scales, ensembles of SVM classifiers for a better global prediction from local predictions, temporal adjustments to improve consistency and accuracy over sequences of predictions, and AVFD for improving the speed of detection. With all of the previously mentioned improvements, it has been shown to have the best cross-dataset performance. On the other hand, FERNet is a compact neural network that achieves a similar accuracy to the state of the art with a significantly faster performance on the same datasets and a better accuracy on new data than *ResNet*.

REFERENCES

[1] D. Matsumoto and B. Willingham, "Spontaneous facial expressions of emotion of congenitally and noncongenitally blind individuals," *J. Personality Social Psychol.*, vol. 96, no. 1, pp. 1–10, 2009.
[2] A. Vinciarelli, M. Pantic, and H. Bourlard, "Social signal processing: Survey of an emerging domain," *Image Vis. Comput.*, vol. 27, no. 12, pp. 1743–1759, Nov. 2009.
[3] P. Lucey, J. Cohn, S. Lucey, I. Matthews, S. Sridharan, and K. M. Prkachin, "Automatically detecting pain using facial actions," in *Proc. 3rd Int. Conf. Affect. Comput. Intell. Interact. Workshops*, Sep. 2009, pp. 1–8.
[4] N. E. Zarif, L. Montazeri, and M. Sawan, "Real-time retinal processing for high-resolution optogenetic stimulation device," in *Proc. 40th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Jul. 2018, pp. 5946–5949.
[5] E. Vural, M. Çetin, A. Erçil, G. Littlewort, M. Bartlett, and J. Movellan, "Automated drowsiness detection for improved driving safety," in *Proc. Int. Conf. Automot. Technol. (ICAT)*, Istanbul, Turkey, 2008.
[6] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685, Jun. 2001.
[7] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image Vis. Comput.*, vol. 47, pp. 3–18, Mar. 2016.
[8] W. T. Meshach, S. Hemajothi, and E. M. Anita, "Real-time facial expression recognition for affect identification using multi-dimensional SVM," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 6, pp. 1–11, 2020.
[9] W.-S. Chu, F. De la Torre, and J. F. Cohn, "Selective transfer machine for personalized facial expression analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 529–545, Mar. 2017.
[10] X. Liu, X. Cheng, and K. Lee, "GA-SVM-based facial emotion recognition using facial geometric features," *IEEE Sensors J.*, vol. 21, no. 10, pp. 11532–11542, May 2020.
[11] M. F. Valstar, T. Almaev, J. M. Girard, G. McKeown, M. Mehu, L. Yin, M. Pantic, and J. F. Cohn, "FERA 2015–second facial expression recognition and analysis challenge," in *Proc. 11th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, May 2015, pp. 1–8.
[12] G. Zhao and M. Pietikäinen, "Dynamic texture recognition using local binary patterns with an application to facial expressions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 915–928, Jun. 2007.
[13] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[14] T. S. Lee, "Image representation using 2D Gabor wavelets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 10, pp. 959–971, Oct. 1996.
[15] S. Agrawal and P. Khatri, "Facial expression detection techniques: Based on viola and jones algorithm and principal component analysis," in *Proc. 5th Int. Conf. Adv. Comput. Commun. Technol.*, Feb. 2015, pp. 108–112.
[16] M. H. Siddiqi, R. Ali, A. M. Khan, E. S. Kim, G. J. Kim, and S. Lee, "Facial expression recognition using active contour-based face detection, facial movement-based feature extraction, and non-linear feature selection," *Multimedia Syst.*, vol. 21, no. 6, pp. 541–555, Nov. 2015.
[17] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2010, pp. 94–101.
[18] A. Swapna, S. Bikash, and P. M. Dipti, "Anubhav: Recognizing emotions through facial expression," *Vis. Comput.*, vol. 34, no. 2, pp. 177–191, Feb. 2018.
[19] M. F. Valstar, E. Sánchez-Lozano, J. F. Cohn, L. A. Jeni, J. M. Girard, Z. Zhang, L. Yin, and M. Pantic, "FERA 2017—Addressing head pose in the third facial expression recognition and analysis challenge," in *Proc. 12th IEEE Int. Conf. Autom. Face Gesture Recognit. (FG)*, May 2017, pp. 839–847.
[20] Z. Sun, Z.-P. Hu, M. Wang, and S.-H. Zhao, "Discriminative feature learning-based pixel difference representation for facial expression recognition," *IET Comput. Vis.*, vol. 11, no. 8, pp. 675–682, Dec. 2017.
[21] M. S. Zia, M. Hussain, and M. A. Jaffar, "A novel spontaneous facial expression recognition using dynamically weighted majority voting based ensemble classifier," *Multimedia Tools Appl.*, vol. 77, pp. 1–31, Oct. 2018.
[22] M. S. Zia and M. A. Jaffar, "Facial expressions recognition using an ensemble of feature sets based on key-point descriptors," *Imag. Sci. J.*, vol. 63, no. 3, pp. 160–167, Mar. 2015.
[23] S. L. Happy and A. Routray, "Automatic facial expression recognition using features of salient facial patches," *IEEE Trans. Affective Comput.*, vol. 6, no. 1, pp. 1–12, Jan. 2015.
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[25] A. Mollahosseini, D. Chan, and M. H. Mahoor, "Going deeper in facial expression recognition using deep neural networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2016, pp. 1–10.
[26] A. Majumder, L. Behera, and V. K. Subramanian, "Automatic facial expression recognition system using deep network-based data fusion," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 103–114, Jan. 2018.
[27] X. Chen, X. Yang, M. Wang, and J. Zou, "Convolution neural network for automatic facial expression recognition," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2017, pp. 814–817.
[28] D. Sen, S. Datta, and R. Balasubramanian, "Facial emotion classification using concatenated geometric and textural features," *Multimedia Tools Appl.*, vol. 78, no. 8, pp. 1–37, 2018.
[29] Z. Yu and C. Zhang, "Image based static facial expression recognition with multiple deep network learning," in *Proc. ACM Int. Conf. Multimodal Interact.*, Nov. 2015, pp. 435–442.
[30] K. Zhang, Y. Huang, Y. Du, and L. Wang, "Facial expression recognition based on deep evolutional spatial-temporal networks," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4193–4203, Sep. 2017.
[31] D. H. Kim, W. J. Baddar, J. Jang, and Y. M. Ro, "Multi-objective based spatio-temporal feature representation learning robust to expression intensity variations for facial expression recognition," *IEEE Trans. Affect. Comput.*, vol. 10, no. 2, pp. 223–236, Apr. 2019.
[32] 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.
[33] V. Mayya, R. M. Pai, and M. M. M. Pai, "Automatic facial expression recognition using DCNN," *Procedia Comput. Sci.*, vol. 93, pp. 453–461, Jan. 2016.
[34] P. Dhankhar, "ResNet-50 and VGG-16 for recognizing facial emotions," *Int. J. Innov. Eng. Technol.*, vol. 13, no. 4, pp. 126–130, 2019.
[35] 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.
[36] P. D. M. Fernandez, F. A. G. Peña, T. I. Ren, and A. Cunha, "FERAtt: Facial expression recognition with attention net," 2019, *arXiv:1902.03284*. [Online]. Available: http://arxiv.org/abs/1902.03284
[37] J. Cai, Z. Meng, A. S. Khan, Z. Li, J. O'Reilly, S. Han, and Y. Tong, "Identity-free facial expression recognition using conditional generative adversarial network," 2019, *arXiv:1903.08051*. [Online]. Available: http://arxiv.org/abs/1903.08051

[38] F. Zhang, T. Zhang, Q. Mao, and C. Xu, "Geometry guided pose-invariant facial expression recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 4445–4460, 2020.
[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] B. Yang, J. Cao, R. Ni, and Y. Zhang, "Facial expression recognition using weighted mixture deep neural network based on double-channel facial images," *IEEE Access*, vol. 6, pp. 4630–4640, 2017.
[41] M. V. Zavarez, R. F. Berriel, and T. Oliveira-Santos, "Cross-database facial expression recognition based on fine-tuned deep convolutional network," in *Proc. 30th SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2017, pp. 405–412.
[42] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE Trans. Affect. Comput.*, early access, Mar. 17, 2020, doi: 10.1109/TAFFC.2020.2981446.
[43] D. A. Pitaloka, A. Wulandari, T. Basaruddin, and D. Y. Liliana, "Enhancing CNN with preprocessing stage in automatic emotion recognition," *Procedia Comput. Sci.*, vol. 116, pp. 523–529, Jan. 2017.
[44] S. Umer, R. K. Rout, C. Pero, and M. Nappi, "Facial expression recognition with trade-offs between data augmentation and deep learning features," *J. Ambient Intell. Humanized Comput.*, pp. 1–15, Jan. 2021.
[45] S. M. González-Lozoya, J. de la Calleja, L. Pellegrin, H. J. Escalante, M. A. Medina, and A. Benitez-Ruiz, "Recognition of facial expressions based on CNN features," *Multimedia Tools Appl.*, vol. 79, pp. 1–21, May 2020.
[46] Z. Yu, G. Liu, Q. Liu, and J. Deng, "Spatio-temporal convolutional features with nested LSTM for facial expression recognition," *Neurocomputing*, vol. 317, pp. 50–57, Nov. 2018.
[47] K. Sikka, G. Sharma, and M. Bartlett, "LOMo: Latent ordinal model for facial analysis in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5580–5589.
[48] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, "From facial expression recognition to interpersonal relation prediction," *Int. J. Comput. Vis.*, vol. 126, no. 5, pp. 550–569, May 2018.
[49] H. I. Dino and M. B. Abdulrazzaq, "Facial expression classification based on SVM, KNN and MLP classifiers," in *Proc. Int. Conf. Adv. Sci. Eng. (ICOASE)*, Apr. 2019, pp. 70–75.
[50] Y. Lv, Z. Feng, and C. Xu, "Facial expression recognition via deep learning," in *Proc. Int. Conf. Smart Comput.*, Nov. 2014, pp. 745–750.
[51] M. Verma, S. K. Vipparthi, and G. Singh, "HiNet: Hybrid inherited feature learning network for facial expression recognition," *IEEE Lett. Comput. Soc.*, vol. 2, no. 4, pp. 36–39, Dec. 2019.
[52] N. Aifanti, C. Papachristou, and A. Delopoulos, "The mug facial expression database," in *Proc. 11th Int. Workshop Image Anal. Multimedia Interact. Services (WIAMIS)*, 2010, pp. 1–4.
[53] C. A. Corneanu, M. O. Simón, J. F. Cohn, and S. E. Guerrero, "Survey on RGB, 3D, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1548–1568, Aug. 2016.
[54] N. E. Zarif. (2020). *Angled Posed Facial Expression Dataset*. [Online]. Available: http://dx.doi.org/10.21227/awgd-3t83
[55] E. Goeleven, R. De Raedt, L. Leyman, and B. Verschuere, "The Karolinska directed emotional faces: A validation study," *Cognition Emotion*, vol. 22, no. 6, pp. 1094–1118, Aug. 2008.
[56] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[57] E. Friesen and P. Ekman, "Facial action coding system: A technique for the measurement of facial movement," Palo Alto, Santa Clara, CA, USA, Tech. Rep., 1978.
[58] S. Han, Z. Meng, A.-S. Khan, and Y. Tong, "Incremental boosting convolutional neural network for facial action unit recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 109–117.
[59] M. Suk and B. Prabhakaran, "Real-time mobile facial expression recognition system—A case study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 132–137.
[60] D. J. Kim, "Facial expression recognition using ASM-based post-processing technique," *Pattern Recognit. Image Anal.*, vol. 26, no. 3, pp. 576–581, Jul. 2016.
[61] O. Rudovic, M. Pantic, and I. Patras, "Coupled Gaussian processes for pose-invariant facial expression recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1357–1369, Jun. 2013.
[62] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, p. 27, 2011.
[63] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, Jan. 2009.
[64] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

NIZAR EL ZARIF received the bachelor's degree in electronics and communications engineering from Beirut Arab University, Lebanon, in 2011, and the master's degree in electrical and computer engineering from the American University of Beirut, Lebanon, in 2015. He is currently pursuing the Ph.D. degree with the Polystim Neurotech Laboratory, Polytechnique Montreal, Canada. His research interests include high-performance computing, machine learning, embedded system design, computer vision, and parallel computing.

LEILA MONTAZERI received the B.S. degree in electrical engineering from Urmia University, Iran, in 2007, and the M.S. degree in electronics from the Noshirvani University of Technology, Iran, in 2012. She is currently pursuing the Ph.D. degree with the Polystim Neurotech Laboratory, Polytechnique Montreal, Canada. Her current research aim is to develop a photostimulator tool for optogenetic vision restoration specific for AMD patients.

FRANÇOIS LEDUC-PRIMEAU (Member, IEEE) received the B.Eng., M.Eng., and Ph.D. degrees in computer engineering from McGill University, Montréal, QC, Canada, in 2007, 2010, and 2016, respectively. He is currently an Assistant Professor with the Department of Electrical Engineering, Polytechnique Montréal, Montréal. His research interests include design and analysis of digital systems, with a focus on telecommunication, signal processing, and machine-learning systems, and on novel approaches for improving the energy efficiency of digital circuits. His work is sponsored by the Data Valorisation Institute (IVADO). He is also a member of the Québec Microsystems Strategic Alliance (ReSMiQ) and the l'Ordre des ingénieurs du Québec (OIQ). He has served as a Vice Co-Chair for the 2021 International Symposium on Topics in Coding (ISTC) and a Track Chair for NEWCAS 2020.

MOHAMAD SAWAN (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Université de Sherbrooke, Canada, in 1990. He is currently a Chair Professor Founder and the Director of the Center for Biomedical Research And INnovation (CenBRAIN), Westlake University, Hangzhou, China. He is also an Emeritus Professor with Polytechnique Montréal. He is an Officer of the National Order of Quebec. He has published more than 800 peer reviewed articles and 12 patents. He has received several awards, including the Queen Elizabeth II Golden Jubilee Medal and the Medal of Merit from the President of Lebanon. He is a fellow of the Canadian Academy of Engineering and fellow of the Engineering Institutes of Canada. Since 2019, he has been the Vice-President Publications of the IEEE CAS Society.