EfficientNet-Lite and Hybrid CNN-KNN Implementation for Facial Expression Recognition on Raspberry Pi

MOHD NADHIR AB WAHAB1, (Member, IEEE), AMRIL NAZIR2, ANTHONY TAN ZHEN REN1, MOHD HALIM MOHD NOOR1, (Member, IEEE), MUHAMMAD FIRDAUS AKBAR3, (Member, IEEE), AND AHMAD SUFRIL AZLAN MOHAMED1

1School of Computer Sciences, Universiti Sains Malaysia, Minden, Penang 11800, Malaysia
2Department of Information Systems and Technology Management, College of Technological Innovation, Zayed University, Abu Dhabi, United Arab Emirates
3School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, Nibong Tebal, Penang 14300, Malaysia

Corresponding authors: Mohd Nadhir Ab Wahab (mohdnadhir@usm.my), Mohd Halim Mohd Noor (halimnoor@usm.my), and Muhammad Firdaus Akbar (firdaus.akbar@usm.my)

This work was supported in part by the Ministry of Higher Education Malaysia for Fundamental Research Grant Scheme under Project FRGS/1/2020/STG06/USM/02/4, and in part by the Universiti Sains Malaysia Short Term Grant 304/PKOMP/6315262.

ABSTRACT Facial expression recognition (FER) is the task of determining a person’s current emotion. It plays an important role in healthcare, marketing, and counselling. With the advancement in deep learning algorithms like Convolutional Neural Network (CNN), the system’s accuracy is improving. A hybrid CNN and k-Nearest Neighbour (KNN) model can improve FER’s accuracy. This paper presents a hybrid CNN-KNN model for FER on the Raspberry Pi 4, where we use CNN for feature extraction. Subsequently, the KNN performs expression recognition. We use the transfer learning technique to build our system with an EfficientNet-Lite model. The hybrid model we propose replaces the Softmax layer in the EfficientNet with the KNN. We train our model using the FER-2013 dataset and compare its performance with different architectures trained on the same dataset. We perform optimization on the Fully Connected layer, loss function, loss optimizer, optimizer learning rate, class weights, and KNN distance function with the k-value. Despite running on the Raspberry Pi hardware with very limited processing power, low memory capacity, and small storage capacity, our proposed model achieves a similar accuracy of 75.26% (with a slight improvement of 0.06%) to the state-of-the-art’s Ensemble of 8 CNN model.

INDEX TERMS EfficientNet-Lite, hybrid CNN-KNN, facial expression recognition, Raspberry Pi, emotion recognition.

I. INTRODUCTION Emotions are natural states associated with the nervous system that influence every aspect of human behaviour, including rationality and decision-making [1], [2]. Individuals can convey emotions through speech, body posture, gestures, and facial expressions.

Facial expressions are effective ways to recognize one’s emotions. Facial expressions are vital for day-to-day communication, as they convey non-verbal emotions and feelings. With just 43 different facial muscles, humans can make 6,000 to 10,000 expressions [3]. In 1872, Charles Darwin hypothesized that humans had evolved facial expressions from animal ancestors. Furthermore, certain expressions are universal across cultures, despite differences in race, language, and religion [4]. In the late 20th century, Ekman and Friesen confirmed Darwin’s theory and classified six universal facial expressions: happy, fear, surprise, disgust, sad, and angry [3].

Facial Expression Recognition (FER) is a computer vision field that uses various techniques to identify emotions from human facial expressions. Researchers are interested in FER, as understanding one’s emotions can improve human-machine interaction, behavioural science, and clinical practice. Recent advancements in computer hardware and image classification techniques allow researchers to develop...
Table 1 summarises the performance of previous research on FER. Yu and Bhamu [5] first attempted to design a FER algorithm that learns features without hand-crafting. Jabid et al. [6] and Shima and Omori [7] improved the algorithm and obtained higher accuracy on the same dataset (i.e., JAFFE) by 90.1% and 95.3%, respectively. The JAFFE dataset is small, as it only contains 213 images. Researchers, therefore, attempted to develop new FER algorithms on large-scale datasets such as the Extended Cohn–Kanade (CK+), comprising 593 images. Shan et al. [8] conducted initial work on the CK+ dataset, and they achieved an accuracy of 95.1%. Mehendale [9] and Breuer and Kimmel [10] improved accuracy to 96% and 98.6%, respectively. The FER-2013 dataset is the most challenging dataset to apply the FER algorithm since it contains 35,887 images. Saeed et al. [11] initially set a baseline accuracy of 68% on the FER-2013 dataset. In 2018, Goodfellow et al. [12] attempted to apply both the Histograms of Oriented Gradients (HOG) feature extractor and Support Vector Machine (SVM) to the FER-2013 dataset. Still, they only achieved 57.7% accuracy, which is worse than the baseline. Recently, Pramerdorfer and Kampel [13] achieved 72.7% accuracy on the FER-2013 dataset using CNN with VGG neural networks.

The neural network’s approach has shown promising performance for FER applications. The method, however, requires the processing power of a high-performance computer. To allow for portability, researchers implemented the FER applications on embedded devices. Sun and An [14] developed a FER system on Linux using HMM as a framework running on an Intel embedded processor, PXA270 and demonstrated satisfactory accuracy. Turabzadeh et al. [15] built a real-time emotion state detection system on FPGA. Loza-Álvarez [16] developed a CNN for FER and applied it to an assistant humanoid robot running on a Raspberry Pi 3.

CNN-based deep learning models achieve the highest accuracy when benchmarked across all the different FER datasets. Moreover, these models provide feature extraction and image classification in a single step compared to feature extraction like HOG or LBP combined with image classification algorithms like SVM or KNN. In image classification applications, hybrid models using a combination of CNN with KNN or the SVM classifier achieved slightly higher accuracies compared to CNN alone. Consequently, researchers have proposed using machine learning and deep learning, such as Support Vector Machine (SVM) or Convolutional Neural Network (CNN). Researchers, however, have not explored this hybrid model in FER applications. The remainder of this paper is organized as follows. Section II presents related work, covering the review of currently developed FER systems. Section III provides the design methodology and architecture of our proposed hybrid CNN-KNN FER model on the Raspberry Pi. Section IV illustrates the practical experimental results and discussion. Finally, Section V contains conclusions and future work.

## II. RELATED WORK

Research has improved the facial expression recognition (FER) algorithm and model performance in the last decade. Table 1 summarises the performance of previous research on FER. Yu and Bhamu [5] first attempted to design a FER algorithm that learns features without hand-crafting. Jabid et al. [6] and Shima and Omori [7] improved the algorithm and obtained higher accuracy on the same dataset (i.e., JAFFE) by 90.1% and 95.3%, respectively. The JAFFE dataset is small, as it only contains 213 images. Researchers, therefore, attempted to develop new FER algorithms on large-scale datasets such as the Extended Cohn–Kanade (CK+), comprising 593 images. Shan et al. [8] conducted initial work on the CK+ dataset, and they achieved an accuracy of 95.1%. Mehendale [9] and Breuer and Kimmel [10] improved accuracy to 96% and 98.6%, respectively. The FER-2013 dataset is the most challenging dataset to apply the FER algorithm since it contains 35,887 images. Saeed et al. [11] initially set a baseline accuracy of 68% on the FER-2013 dataset. In 2018, Goodfellow et al. [12] attempted to apply both the Histograms of Oriented Gradients (HOG) feature extractor and Support Vector Machine (SVM) to the FER-2013 dataset. Still, they only achieved 57.7% accuracy, which is worse than the baseline. Recently, Pramerdorfer and Kampel [13] achieved 72.7% accuracy on the FER-2013 dataset using CNN with VGG neural networks.

The neural network’s approach has shown promising performance for FER applications. The method, however, requires the processing power of a high-performance computer. To allow for portability, researchers implemented the FER applications on embedded devices. Sun and An [14] developed a FER system on Linux using HMM as a framework running on an Intel embedded processor, PXA270 and demonstrated satisfactory accuracy. Turabzadeh et al. [15] built a real-time emotion state detection system on FPGA. Loza-Álvarez [16] developed a CNN for FER and applied it to an assistant humanoid robot running on a Raspberry Pi 3.

CNN-based deep learning models achieve the highest accuracy when benchmarked across all the different FER datasets. Moreover, these models provide feature extraction and image classification in a single step compared to feature extraction like HOG or LBP combined with image classification algorithms like SVM or KNN. In image classification applications, hybrid models using a combination of CNN with KNN or the SVM classifier achieved slightly higher accuracies compared to CNN alone. Consequently, researchers have proposed using machine learning and deep learning, such as Support Vector Machine (SVM) or Convolutional Neural Network (CNN). Researchers, however, have not explored this hybrid model in FER applications. The remainder of this paper is organized as follows. Section II presents related work, covering the review of currently developed FER systems. Section III provides the design methodology and architecture of our proposed hybrid CNN-KNN FER model on the Raspberry Pi. Section IV illustrates the practical experimental results and discussion. Finally, Section V contains conclusions and future work.

| Algorithm | Accuracy (%) | Dataset Tested |
|-----------|--------------|----------------|
| Gabor Filter + SVM [5] | 80.9 | JAFFE (213 images) |
| LBP [6] | 90.1 | JAFFE (213 images) |
| CNN + SVM [7] | 95.3 | JAFFE (213 images) |
| LBP + SVM [8] | 95.1 | CK+ (593 images) |
| CNN [9] | 96.0 | CK+ (593 images) |
| CNN [10] | 98.6 | CK+ (593 images) |
| Human Accuracy [11] | 68.0 | FER-2013 (35,887 images) |
| HoG + SVM [12] | 57.7 | FER-2013 (35,887 images) |
| CNN with VGG [13] | 72.7 | FER-2013 (35,887 images) |
accuracy than the standard CNN models [20], [21], [22], [23]. According to the literature we examined, researchers have not applied this approach to FER, and there is little work regarding embedded devices. Additionally, existing works’ low accuracy for the FER-2013 dataset illustrates room for accuracy improvement. We present a lightweight approach that runs efficiently and attains higher accuracy on the Raspberry Pi.

III. METHODOLOGY
We outline the design methodology we used to develop the hybrid CNN-KNN FER model. Furthermore, we explain the methods to develop the FER system with acceptable accuracy and performance on embedded devices.

Subsection A describes the design technique and steps involved in developing the hybrid CNN-KNN model. KNN is thought to be merged with CNN since it is simple and easy to use, potentially conducting the training phase quickly and at no cost [24]. Subsection B provides a high-level summary of our suggested FER technique. Finally, Section C describes the FER dataset that was utilized for training.

A. DESIGN PROCEDURE
The steps to design the FER system are:
(i) Determine the features needed for the FER system.
(ii) Select and prepare the FER dataset.
(iii) Design and develop a CNN-KNN model for FER.
(iv) Code and train the model with the dataset on TensorFlow using Google Colab.
(v) Evaluate the accuracy of the model.
(vi) Convert the model to TensorFlow lite for Raspberry Pi.
(vii) Develop and code pre-processing image methods for real-time webcam video on Raspberry Pi.
(viii) Test performance of FER in terms of inferencing time on the Raspberry Pi.

B. FEATURES
The features of the FER system is:
- Identify seven expressions: angry, disgust, happy, sad, fear, surprise, and neutral.
- Training and evaluation were done on the FER-2013 dataset.
- CNN model for feature extraction and KNN for expression recognition.
- On the Raspberry Pi, a real-time FER programme with a webcam.

C. FER SYSTEM ARCHITECTURE
Fig. 1 shows the system consists of the webcam as input, a Raspberry Pi 4 controlling the FER system, and a display monitor to show the predicted expression results. We chose the Raspberry Pi 4 because it supports TensorFlow, which is inexpensive and has sufficient computing power for the neural network model.

The development and research of this system consist of four parts:
- Facial expression dataset
- Image pre-processing
- CNN model for feature extraction
- KNN classifier for expression recognition

1) FACIAL EXPRESSION DATASET
We trained the CNN-KNN model using 35,887 static grayscale images from the FER-2013 dataset [19]. The dataset contains seven facial expressions we collected from the real world with various faces of different ages and facial...
orientations. The size of each image is 48 pixels × 48 pixels. Table 2 shows the distribution of the training images in the FER-2013 dataset. Naturally, the distribution of images varies. The happy expression has the highest number with 8110 images, and the disgust expression has the lowest number with 492 images. The happy expression has the most images in the training dataset, with 25.11 percent of the distribution number.

Table 2. Distribution of Training Images in FER-2013 Dataset.

| Expression | Number of Images | % of distribution |
|------------|-----------------|-------------------|
| Angry      | 4462            | 13.82%            |
| Disgust    | 492             | 1.52%             |
| Happy      | 8110            | 25.11%            |
| Sad        | 5483            | 16.98%            |
| Fear       | 4593            | 14.22%            |
| Surprise   | 3586            | 11.10%            |
| Neutral    | 5572            | 17.25%            |

In contrast, the disgust expression has the fewest images, with 1.52 percent of the distribution number. Table 3 shows the distribution of testing images for evaluating the CNN-KNN model. Similar to the training data, the happy expression has the highest number of images available in the test set with 24.49 percent of the distribution number. The disgust expression has the lowest percentage in the test set with only 1.53 percent. In conclusion, both the train and test distributions are similar. Fig. 2 shows sample images from the FER-2013 dataset. We chose this dataset because it has the most images for all appropriate facial expressions, and it provides labelled grayscale images with cropped faces. Furthermore, the FER-2013 dataset is publicly available with various models for comparison. The dataset images also represent realistic conditions with variations in age, race, and pose.

Artificially modifying the current images can increase the dataset for training using the image augmentation technique. In image augmentation, one can create new images for the training dataset from the original dataset by applying image flipping, rotation, scaling, or adding noise to the original image in the dataset.

2) IMAGE PRE-PROCESSING

As part of the image pre-processing for this study, we converted the frames captured with a webcam connected to the Raspberry Pi from RGB to grayscale, as shown in Fig. 3. The Haar-Cascade classifier, chosen for its low computing cost and reasonable accuracy, next detected the faces then cropped and resized the face. Python’s OpenCV library implemented the Haar-Cascade classifier and performed the rescaling, while Python Imaging Library (PIL) converted the image to grayscale. Minimal pre-processing ensures real-time capability. All the image pre-processing performed on the Raspberry Pi. The dataset is used as it is without any augmentation process to balance the data because we aim to see the behaviour of the proposed method on this imbalanced dataset.

3) FEATURE EXTRACTION AND CLASSIFICATION

In the CNN model for FER, feature extraction derived important information from an image, differentiating between expressions. Convolutional layers, pooling layers, and activation functions performed feature extraction. KNN classifier is used for expression recognition based on the features extracted from CNN.
We chose CNN as the algorithm for feature extraction from the literature review of FER systems because it provides the best accuracy. Since no FER dataset with millions of images exists, training on a limited dataset might not yield high accuracy. Thus, transfer learning solves the problem of insufficient training samples while maintaining accuracy.

The CNN model for transfer learning is the EfficientNet Model. EfficientNet has a reputation for achieving high accuracy with minimal parameters and FLOPS (Floating Point Operations Per Second). It is suitable for use with the Raspberry Pi, which has limited processing power. We implemented transfer learning using the learned weights of EfficientNet from the ImageNet dataset since both FER-2013, and the ImageNet are image classification datasets.

Inverted Residual Block (MBConv), like MobileNetV2, is the building block in EfficientNet. Unlike traditional CNNs, which involve manual fine-tuning of three dimensions: number of layers (depth scaling), number of channels (width scaling), and image size (resolution scaling), EfficientNet uses the model compound scaling process to scale up the CNN. Moreover, EfficientNet uses a Swish activation function differing from the normal ReLU function found in the conventional CNN model. The Swish function is a multiplication of a linear and a sigmoid activation [18]. Table 4 shows the EfficientNet CNN topology. The input image size (48 x 48) is resized to the standard CNN input layer (224 x 224), which are standard practice for varying input data size [25]–[27].

The CNN performed the feature extraction on the input images in various stages. In CNN’s feature extraction, the architecture consisted of 7 inverted residual blocks (MBConv) and two residual blocks (Conv). Fig. 4 and Table 4 show a complete workflow of MBConv1, k3 x 3, and MBConv6, k3 x 3 block. Both MBConv1, k3 x 3, and MBConv6, k3 x 3 use depthwise convolution, which integrates a kernel size 3 x 3 with a stride size of s. Batch Normalization, activation, and convolution are included in these two blocks, which have a 1 x 1 kernel size. The classifier and the expression prediction are the two stages of the KNN classification. The KNN took the place of the Softmax and the traditional pooling layer in the final image categorization.

KNN is a suitable algorithm for multiclass classification problems. Hence, we proposed a hybrid CNN-KNN model, as KNN provides better accuracy as a classifier, especially in a noisy environment [17]. Since researchers have not evaluated the hybrid CNN-KNN model in the FER framework, we use this hybrid model to improve accuracy in the FER-2013 dataset. In the Hybrid CNN-KNN model, the KNN classifier replaced the pooling and Softmax later at the output of EfficientNet.

4) MODEL TRAINING AND EVALUATION
We used Python to build the FER model since Python supports Raspberry Pi and most deep learning frameworks. We used TensorFlow as the deep learning framework since it is the most popular framework. Moreover, it has a lite version (TensorFlow Lite) to support mobile and edge devices like Raspberry Pi. The TensorFlow Lite model also performs better than the regular TensorFlow model.

Our group used Google Colab notebooks for training and evaluation, and we executed these notebooks in the browser using Google’s cloud server. Google Colab is free and utilizes Google’s GPU and TPU for training.

Investigators can reuse weights learned, and they can unfreeze some layers of the CNN to perform training, thanks to the use of transfer learning to develop the FER system. Adam optimizer, an adaptive learning rate method for stochastic gradient descent, accomplishes the training. Furthermore, we used a batch size of 32 and a maximum of 100 epochs. In KNN training, we used the default k value of 5 and default Euclidean distance for classification. After that, the optimizer will do the hyperparameter tuning to determine the best k value and distance metric to be used.

To evaluate the training, we used the Stratified k-fold Cross-Validation method with a k-value of 5. The value 5 is
FIGURE 4. Proposed CNN-KNN Model.

FIGURE 5. Sample confusion matrix.

TABLE 5. Comparison of Facial Expression Recognition Model for FER-2013 Dataset.

| Algorithm       | Accuracy (%) |
|-----------------|--------------|
| Proposed Method | 75.26        |
| Ensemble of 8 CNN [13] | 75.20    |
| CNN with VGG [13] | 72.70    |
| HoG + SVM [12]   | 57.70        |
| Human Accuracy [11] | 68.0     |

considered based on our trial-and-error approach. Several $k$ values are considered 1, 2, 3, 4, 5, 7 and 10. However, there is no significant difference when the value is over 5; therefore, 5 is considered. Cross-validation provides robust estimates of the variance of the training data. A confusion matrix (as shown in Fig. 5), averaged to obtain the FER system’s accuracy, evaluated each class’s accuracy (as shown in Fig. 5). We compared our model’s accuracy to other models (shown in Table 5) trained on the FER-2013 dataset. Despite deploying the hybrid CNN-KNN model on the Raspberry Pi hardware with very limited processing power, low memory capacity, and small storage capacity, we achieved similar accuracy to the state-of-the-art’s Ensemble of 8 CNN model with X time speed up of inference time.

Other evaluation methods for the CNN models are sensitivity, specificity, and F1-Score, calculated based on the formulas below:

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \tag{1}
\]
\[
\text{Specificity} = \frac{TN}{TN + FP} \tag{2}
\]
\[
F1 \text{ Score} = \frac{TP}{TP + 0.5 \times (FP + FN)} \tag{3}
\]

After training, we fine-tuned the deep learning model to determine the best accuracy, and TensorFlow provided model evaluation. We translated the TensorFlow model to TensorFlow Lite and copied it to a Raspberry Pi. Additionally, we used inference time, the time it takes the deep learning algorithm to process the image and make predictions, as an evaluation metric. The FER system requires a shorter inferencing time to operate in real-time. The system operated on a Raspberry Pi 4 using a webcam as input.

IV. RESULTS AND DISCUSSION

We present the results and discussions regarding the training and implementation of the hybrid CNN-KNN FER model. Additionally, we present experimental results to improve the accuracy during the training of the FER model. We then perform optimization of the parameters of the model. Finally, we present the model’s final performance regarding the accuracy, sensitivity, specificity, and inference time.

A. TRAINING AND EVALUATION OF CNN MODEL/ZZZZZ/WWW/IMAGE PRE-PROCESSING AND DATA AUGMENTATION

Our group executed the training and evaluation of the CNN model on the Google Colab platform. TensorFlow 2.4 on Python 3.6 and Nvidia V100 GPU accelerated the training. Moreover, Stratified 5-fold Cross-Validation validated all training. Stratified 5-fold Cross-Validation ran two times, and it recorded the highest accuracy from the two runs.

Before training the CNN model, image pre-processing resized the images to the required shape, while data augmentation techniques increased the sample size. Table 6 shows the data augmentation methods and rescaling used to ensure...
images have input range from −1.0 to 1.0. The ImageDataGenerator class in the Keras library performed the data augmentation and image pre-processing.

We used EfficientNet as the model for transfer learning. It has several versions with different parameters, and it supports different usages. The number of trainable weights can affect the accuracy and inference time of the model. We used EfficientNet-Lite L0-L4 models in this FER application instead of the full EfficientNet B0-B7 models as the EfficientNet-Lite is optimized for edge devices. EfficientNet-Lite removes squeeze-and-excitation networks, and it replaces swish activation functions with ReLU6 activation to support the quantization needed for edge devices.

EfficientNet-Lite models train on the FER-2013 dataset with images resized to the required size of 224 × 224. Fig. 6 demonstrates that each model uses the same architecture. Furthermore, we tested all models using the same settings in Table 7 with the same image pre-processing method in Table 6. Table 8 compares the test accuracy of the 3589 test images and inference time on Colab.

From Table 8, the EfficientNet-Lite L3 model has the best test accuracy, but its drawback is a high inference time. Since the difference in accuracy between models is minimal, we selected the L0 model to have the fastest inference time. The EfficientNet-Lite L0 model provides a test accuracy of 67.79%, and it has the fastest inference time of 540.9ms. It also has the least number of weights to train the models.

In Fig. 7, the accuracy for the validation set stabilizes at five epochs. One can see overfitting after this point. 16 epochs provide the best accuracy. Moreover, the validation accuracy starts higher than the training accuracy, indicating this test data consists of “easier” examples than the train set. Fig. 8 also illustrates the model loss with increasing epochs. We observed the train losses steadily decrease due to the overfitting, while the validation loss suddenly increases at epoch 14.

With the FER-2013 dataset as the only training data, the maximum test accuracy we achieved with the EfficientNet-Lite L0 model was 67.79%. Investigators can increase the number of training images to improve accuracy. As a result, we supplemented the FER-2013 training dataset with additional FER training data from the JAFFE and KDEF databases. The updated model training distribution is shown in Table 9. The testing dataset for FER was kept the same as the original FER-2013 dataset for benchmarking purposes. With the additional training data and the same setup as Table 7, the L0 model’s test accuracy increased from 67.79% to 69.21%.
From Fig. 9, the Fully Connected layer for classification has a Global Average Pooling layer, a Dropout layer, and a Dense layer with Softmax activation function to recognize expressions. To improve accuracy and prevent overfitting, we explored a deeper Fully Connected layer combined with Dropout layers, the Dense layer, and the Batch Normalization layer.

In Fig. 10, one sees overfitting as validation loss starts to climb higher than training loss at the 5th epoch. We increased the Dropout layer’s drop rate to prevent overfitting, and a few Dropout layers were used. The drop rate of the Dropout layer is the probability a node is enabled for weight optimization during training. The Batch Normalization layer helps standardize the inputs, reduce the generalization error, and improve the training speed.

No correct method exists to determine the best network design. We built the Fully Connected layer based on trial and error from the existing CNN model’s Fully Connected layer designs. Although the KNN classifier replaced the Fully Connected layer in the hybrid FER model, we had to optimize the KNN classifier to improve the weight training in the EfficientNet-Lite L0 model. We tested the Fully Connected layer designs with the same settings as Table 7 and the extra training data. Fig. 11 shows the final design of the network with the best accuracy. This model improved test accuracy from 69.21% to 71.05%. Appendix A details the full architecture of the EfficientNet-Lite L0 model.

In CNN, the loss function compares the current model’s error to the training result by calculating weights’ errors. The function minimizes the error of the model as much as possible.

We tested the model from Fig. 11 with different loss functions with default parameters, and Table 11 shows the test accuracy of each loss function. Focal loss and CEFL2 loss are new loss functions to improve the imbalanced class datasets. The Adam optimizer uses the same settings as Table 11 to
test each loss function. The system’s accuracy has improved significantly as a result of the design change, as seen in Fig. 12. With these enhancements, we were able to alleviate the problem of overfitting. Furthermore, we observed a slight improvement in accuracy, as both training and validation accuracies are similar. Finally, we observed a consistent improvement at the 35th epoch with more than 69% accuracy.

Fig. 13 shows the training and validation loss with Cross-Entropy vs epochs. Based on Table 12, this model’s best loss function is the CEFL2 loss, which improved test accuracy to 71.89%. The CEFL2 loss function is:

$$CEFL2 = -\frac{(1-p)^2}{(1-p)^2 + p^2} \log \log p$$

$$-\frac{p^2}{(1-p)^2 + p^2} \log(1-p)^\gamma \log p \quad (4)$$

where $p$ is the ground truth output from the model and hyper-parameter, $\gamma = 2$.

The optimizer calculates and updates the weights in the model based on the loss function output. Table 11 uses CEFL2 loss as the loss function to show the test accuracy compared to different optimizers with default parameters. The best optimizer for this FER model is the Adam optimizer, which produced an accuracy of 71.89%.

We used the optimizer’s learning rate to set the number of updates the weights receive during the model’s training. A high learning rate will reduce the training loss faster, but this high rate may cause a model to converge to a less than optimal solution. A low learning rate may require considerable time to train along with a large number of epochs.
TABLE 13. CNN Model Confusion Matrix on Test Dataset without Class Weight.

|       | Angry | Disgust | Fear | Happy | Neutral | Sad | Surprise |
|-------|-------|---------|------|-------|---------|-----|----------|
| Angry | 329   | 7       | 42   | 11    | 50      | 44  | 8        |
| Disgust | 9 | 38   | 5    | 1     | 1       | 1   | 0        |
| Fear    | 44   | 2      | 282  | 23    | 51      | 80  | 46       |
| Happy   | 9    | 0      | 14   | 800   | 27      | 11  | 18       |
| Neutral | 28   | 1      | 19   | 24    | 492     | 52  | 10       |
| Sad     | 54   | 0      | 47   | 21    | 90      | 377 | 5        |
| Surprise | 5   | 0      | 25   | 17    | 15      | 5   | 349      |

For the experiment, we executed training with a learning rate reducer. It reduced the learning rate once it neared the optimal solution for further improvement. The Adam optimizer tested the learning rate from 0.0001 to 0.1 with the CEFL2 loss function. Based on Table 12, the 0.001 learning rate results in better overall accuracy and finishes at epoch 40. Moreover, Fig. 11 shows the model’s training accuracy and validation accuracy steadily increase up to 25 epochs.

Conversely, the validation accuracy increases until ten epochs before it becomes stagnant. Like previous observations, the validation accuracy starts higher than the training accuracy, possibly indicating that the test data consist of “easier” examples than the training set. The validation accuracy surpasses 70% at 10 epochs and then fluctuates after this point.

This FER model’s test and training dataset have an imbalanced class where some expressions are lower than others. Including class weights helped with the imbalanced FER-2013 training dataset, which had fewer samples for certain expressions, like disgust. Table 13 shows the confusion matrix of the model without class weights used in the training process.

Fig. 12 demonstrates that overfitting is a problem of this model, as validation loss becomes higher than training loss at the 10th epoch. We used class weights during training to solve this problem. The values for class weight are shown in Table 14. With Adam optimizer and a learning rate of 0.001, we tested with extra training data and CEFL2 loss. We examined the performance of the dataset with the uneven distribution using F1-Score. The F1-Score assesses the sensitivity-to-recall ratio. The values for class weight are shown in Table 14. With Adam optimizer and a learning rate of 0.001, we tested with extra training data and CEFL2 loss. We examined the performance of the dataset with the uneven distribution using F1-Score. The F1-Score assesses the sensitivity-to-recall ratio. We calculated F1-Score for each expression and used the weighted-average F1-Score for comparison.

Table 16 shows the performance of test accuracy and weighted average in the F1-Score based metric. The test accuracy score improved from 74.31% without class weight to 74.67% with class weight added. Moreover, when we added the class weights, the weighted-average F1-Score improved from 0.7402 to 0.7456. Fig. 14 shows the validation loss does not start to overfit until it reaches higher validation accuracy.

The final CNN Model, based on EfficientNet-Lite L0, has a test accuracy of 74.67%. Table 15 shows the confusion matrix for the final model with class weights.
for the final CNN model. Table 17 shows the settings we used to achieve the accuracy of 74.67% from the results of the experiments. Table 18 shows the accuracy that each fold in the 5-fold Cross-Validation achieved. Each fold uses 38288 images for training and validation. The inference time for the CNN model in Colab is 716.2ms.

Table 19 shows the final CNN Model, based on EfficientNet-Lite L0. The CNN model has 74.67% accuracy, still lower than the benchmark of 75.2%. We converted
M. N. Ab Wahab et al.: Efficientnet-Lite and Hybrid CNN-KNN Implementation for FER on Raspberry Pi

FIGURE 15. CNN Model with Class Weight Training and Validation Loss vs Epoch.

TABLE 19. Performance evaluation of the final CNN model.

| Performance                  | Value     |
|------------------------------|-----------|
| Test Accuracy                | 74.67%    |
| Weighted-average Sensitivity | 0.7467    |
| Weighted-average Specificity | 0.9375    |
| Weighted-average F1-Score    | 0.7456    |

TABLE 20. Comparison of Tensorflow Vs Tensorflow Lite Models.

| Model          | Test Accuracy (%) | Inference Time on Raspberry Pi (ms) |
|----------------|-------------------|------------------------------------|
| TensorFlow     | 74.67             | 326.91                             |
| TensorFlow Lite| 74.50             | 6.39                               |

the CNN model to the TensorFlow Lite Model, which is optimized for edge devices like Raspberry Pi. Post-training integer quantization optimizes the TensorFlow Lite model, reducing the model size by 75% with a trade-off of lower test accuracy.

Table 20 further shows a performance comparison of TensorFlow and TensorFlow Lite. The TensorFlow has a 0.17% improvement of accuracy compared to the TensorFlow Lite. However, the TensorFlow Lite has a $51 \times$ faster inference time compared to the standard TensorFlow on the Raspberry Pi. While achieving a remarkable improvement in inference time, a small reduction in test accuracy is indispensable for a model running on mobile devices.

B. TRAINING AND EVALUATION OF KNN CLASSIFIER

We performed training of the KNN classifier on Google Colab with TPU. Fig. 16 demonstrates the removal of parts of the Fully Connected layer to train the KNN classifier. Our group froze the weights of the CNN model to generate features like training input for the KNN classifier. We then used the same training and testing data to train the KNN classifier. Furthermore, we used the TensorFlow Lite model with quantization for the CNN. After generating features from the CNN model, training produced a standard scaler to scale the feature input of the KNN classifier in the range 0 to 1. We tested the KNN distance functions to find the best KNN distance function and optimal K value. Table 21 compares test accuracy to a variety of other types of distance functions. Euclidean distance with a k-value of 17 is the KNN distance function with the highest test accuracy, improving the model’s accuracy from 74.67% to 75.26%. Manhattan distance has the largest k-value of the KNN distance function, with a value of 24, achieving 75.15% percent accuracy. The lowest k-value of the KNN distance function is Canberra distance, which also produces a good result with 75.20% accuracy.

Table 21. Comparison of Test Accuracy with different KNN Distance Function.

| KNN Distance Function | Test Accuracy (%) | k-Value |
|-----------------------|-------------------|--------|
| Euclidean             | 75.26             | 17     |
| Manhattan             | 75.15             | 24     |
| Chebyshev             | 74.14             | 12     |
| Hamming               | 74.76             | 13     |
| Canberra              | 75.20             | 7      |
| Braycurtis            | 74.95             | 22     |

The final hybrid CNN-KNN model in Fig. 17 consists of the EfficientNet-Lite L0 model from transfer learning, a Global Average Pooling layer, and a KNN classifier Euclidean distance algorithm. Table 22 shows the confusion matrix for the hybrid CNN-KNN FER model. Table 23 shows the final parameters for the hybrid CNN-KNN Model.

Table 24 demonstrates that the test accuracy of the hybrid CNN-KNN FER model improved by 0.6% compared to using Softmax as the output layer. Using KNN, the proposed model’s accuracy was 0.1% higher than the state-of-the-art FER model, which uses an ensemble of 8 CNN models [13].

C. ANALYSIS ON RASPBERRY PI

Fig. 18 shows the Raspberry Pi FER system. The Raspberry Pi 4 is connected to the webcam, and the monitor displays the system’s output. A connected Coral USB Accelerator
enhances the inferencing of the TensorFlow Lite models through the Edge TPU.

Fig. 19 and Fig. 20 show the seven expressions the Raspberry Pi FER application captures: angry, disgust, fear, happy, neutral, sad, and surprise. We used the Haar-Cascade classifier for face detection, and the CNN-KNN model predicted all the expressions.

Table 25 shows the comparison of Tensorflow, TensorFlow Lite, and our KNN + TensorFlow Lite implementations. It can be seen that our KNN + TensorFlow Lite implementation achieves the best test accuracy of 75.26%.

Table 26 further shows the accuracy comparison for our proposed hybrid FER model on the Raspberry Pi 4 compared
to the state-of-the-art models. Our proposed hybrid CNN-KNN model achieves 75.26% accuracy, which is slightly better than the state-of-the-art Ensemble of 8 CNN with 75.2% accuracy. Moreover, Table 27 compares inference time for the FER model on the Raspberry Pi 4. Post-training quantization optimizes the TensorFlow Lite model with the
TABLE 25. Comparison of TensorFlow, TensorFlow Lite, and our KNN + TensorFlow Lite Models.

| Model                    | Test Accuracy (%) |
|--------------------------|-------------------|
| TensorFlow               | 74.67             |
| TensorFlow Lite          | 74.50             |
| KNN + TensorFlow Lite    | 75.26             |

TABLE 26. Comparison of TensorFlow, TensorFlow Lite, and our KNN + TensorFlow Lite Models.

| Model                                      | Test Accuracy (%) |
|--------------------------------------------|-------------------|
| Proposed Hybrid CNN-KNN                   | 75.26             |
| Model                                      |                   |
| Ensemble of 8 CNN [13]                     | 75.2              |
| CNN with VGG [13]                          | 72.7              |
| HoG + SVM [12]                             | 57.7              |
| Human Accuracy [11]                        | 68.0              |

TABLE 27. Comparison of CNN Model in terms of Inference Time.

| Model                        | Test Accuracy (%) | Inference time on Raspberry Pi (ms) |
|------------------------------|-------------------|-------------------------------------|
| TensorFlow                   | 74.67             | 326.91                              |
| TensorFlow Lite              | 74.50             | 6.39                                |
| KNN + TensorFlow Lite        | 75.26             | 74.15                               |

Softmax output layer. It provides the best inference time among the models. The proposed CNN-KNN model requires a longer inference time due to the KNN classifier, but the inference time is still acceptable.

It is also notable that the performance of our proposed method is better than the shallow CNN by [25] for the same dataset (FER2013), including the comparison methods (AlexNet, HOG+CNN, Xception, VGG-8, FaceLiveNet) where their accuracy results vary between 61-69%.

V. CONCLUSION

In conclusion, we have tested a new hybrid CNN-KNN model for FER. We discussed image pre-processing and data augmentation techniques. Aside from data augmentation to increase the sample size, we combined extra training data from JAFFE and KDEF with the FER-2013 training dataset. Additionally, we discussed optimizing the Fully Connected layer, loss function, optimizer, learning rate, class weights, KNN distance function, and KNN k-value. A hybrid model using CNN for feature extraction and KNN as the classifier can improve FER model accuracy on the FER-2013 dataset. The hybrid CNN-KNN model produced an accuracy of 75.3%, a 0.6% improvement from the CNN model and a 0.1% improvement in accuracy compared to state-of-the-art FER models. The proposed model has a sensitivity of 0.7526, specificity of 0.9393, and inference time on the Raspberry Pi 4 is 74.15ms.

Investigators have developed FER systems with different feature extraction techniques, such as LBP or Gabor Filter, combined with traditional machine learning like SVM or KNN and deep learning models like CNN. While many FER systems on the FER-2013 dataset have good accuracy, the main goal is improving the existing models and evaluating the embedded device’s performance.

We researched methods to recognize facial expressions with deep learning models using embedded devices. Training a hybrid CNN-KNN model for FER, using CNN for feature extraction and KNN for expression recognition, can achieve this goal. We based our model on the EfficientNet model and benchmarked on the FER-2013 dataset. Finally, we compared the different pre-trained EfficientNet models, and we selected the most suitable model for the FER.

We have also shown the KNN classifier can improve the accuracy of the CNN model. Our implementation of the hybrid model on the Raspberry Pi, with webcam and Coral USB Accelerator for model inferencing, demonstrated improved accuracy. The system has a reasonable inference time of 74.15ms when tested on the Raspberry Pi and test accuracy of 75.3%, which is a 0.1% improvement on the state-of-the-art model accuracy and improvement of 0.6% compared to the CNN model without the KNN classifier.

The proposed FER model produced a reasonable accuracy and inference time on the Raspberry Pi. Further research on the proposed CNN-KNN FER model’s accuracy can experiment with more Fully Connected layer designs. Moreover, further research can try different Batch Normalization arrangements and Dropout layer designs. Increasing the number of neurons for the Dense layer in the Fully Connected layer can improve accuracy. Combining more FER datasets can create a larger sample size for training. With limited time to train the model on the Colab platform, we cannot use more training data as there will be insufficient time for the training. We can improve the tuning process of parameters by sweeping through a range of possible values to find the optimal values, instead of manually selecting the value to test.

Further research can perform the model’s testing and benchmark on different FER datasets, as the FER-2013 dataset contains a few misclassifications of the images. We could also discuss on the performance comparison of the proposed method against the variants of CNN itself such as Fast-RNN, Faster-RNN, YOLO and SSD. Considering the Haar-Cascade classifier is used in this application, we could explore a more sophisticated face detection algorithm limited to only frontal faces. Since wearing face masks has become a norm during COVID-19, future work can explore a FER dataset focusing on features of the eyes.

REFERENCES

[1] J. Panksepp, Affective Neuroscience: The Foundations of Human and Animal Emotions. London, U.K.: Oxford Univ. Press, 2004.
[2] C. Izard, The Psychology of Emotions. New York, NY, USA: Springer, 1991.
[3] P. Ekman, and W. Friesen, Facial Action Coding System, vol. 1. Palo Alto, CA, USA: Consulting Psychologists Press, 1978.
[4] N. Carlson, Physiology of Behavior. London, U.K.: Pearson, 2012.
[5] J. Yu and B. Bhanu, “Evolutionary feature synthesis for facial expression recognition,” Pattern Recognit. Lett., vol. 27, no. 11, pp. 1289–1298, Aug. 2006.
[6] T. Jahid, M. H. Kabir, and O. Chae, “Robust facial expression recognition based on local directional pattern,” ETRI J., vol. 32, no. 5, pp. 784–794, Oct. 2010.

[7] Y. Shima and Y. Omori, “Image augmentation for classifying facial expression images by using deep neural network pre-trained with object image database,” in Proc. 3rd Int. Conf. Robot., Control Autom., 2018, pp. 140–146.

[8] C. Shan, S. Gong, and P. W. McOwan, “Facial expression recognition based on local binary patterns: A comprehensive study,” Image Vis. Comput., May 2009, vol. 27, no. 6, pp. 803–816.

[9] N. Mehendale, “Facial emotion recognition using convolutional neural networks (FERC),” Social Netw. Appl. Sci., vol. 2, no. 3, pp. 1–8, Mar. 2020.

[10] R. Breuer and R. Kimmel, “A deep learning perspective on the origin of facial expressions,” 2017, arXiv:1705.01842. [Online]. Available: http://arxiv.org/abs/1705.01842.

[11] S. Saeed, J. Baber, M. Bakhtyar, I. Ullah, N. Sheikh, I. Dad, and A. Ali, “Empirical evaluation of SVM for facial expression recognition,” Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 11, pp. 670–673, 2018.

[12] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D. H. Lee, and Y. Zhou, “Challenges in representation learning: A report on three machine learning contests,” in Proc. Int. Conf. Neural Inf. Process., Nov. 2013, pp. 117–124.

[13] C. Pratendorfer and M. Kampel, “Facial expression recognition using convolutional neural networks: State of the art,” 2016, arXiv:1612.02903. [Online]. Available: http://arxiv.org/abs/1612.02903.

[14] Y. Sun and Y. An, “Research on the embedded system of facial expression recognition based on HMM,” in Proc. 2nd IEEE Int. Conf. Inf. Manage. Eng., Chengdu, China, Apr. 2010, pp. 727–731.

[15] S. Turabzadeh, H. Meng, R. M. Swash, M. Pleva, and J. Juhar, “Real-time emotional state detection from facial expression on embedded devices,” in Proc. 7th Int. Conf. Innov. Comput. Technol. (INTECH), Luton, U.K., Aug. 2017, pp. 46–51.

[16] A. Loza-Álvarez, A. Monroy-Meza, R. A. Suárez-Rivera, G. I. Pérez-Soto, L. A. Morales-Hernández, and K. A. Camarillo-Gómez, “Facial expressions recognition with CNN and its application in an assistant humanoid robot,” in Proc. COMRob. Ensenada, Mexico, Sep. 2018, pp. 1–6.

[17] A.-J. Gallego, A. Pertusa, and J. Calvo-Zaragoza, “Improving convolutional neural networks’ accuracy in noisy environments using k-nearest neighbors,” Appl. Sci., vol. 8, no. 11, pp. 2086, Oct. 2018.

[18] M. Tan and Q. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” in Proc. 36th Int. Conf. Mach. Learn., May 2019, pp. 6105–6114.

[19] FER-2013 Dataset. Accessed: May 1, 2020. [Online]. Available: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data.

[20] B. Srinivas and G. Rao, “A hybrid CNN-KNN model for MRI brain tumor classification,” Int. J. Recent Technol. Eng., vol. 8, no. 2, pp. 20–25, 2019.

[21] K. R. Baheem and I. H. Ali, “Facial expression recognition using hybrid CNN-SVM technique,” Int. J. Adv. Sci. Technol., vol. 29, no. 4, pp. 5528–5534, Jun. 2020.

[22] A.-J. Gallego, J. Calvo-Zaragoza, and J. R. Rico-Juan, “Insights into efficient k-nearest neighbor classification with convolutional neural codes,” IEEE Access, vol. 8, pp. 99312–99326, 2020, doi: 10.1109/ACCESS.2020.2997387.

[23] X. Sun, J. Park, K. Kang, and J. Hur, “Novel hybrid CNN-SVM model for recognition of functional magnetic resonance images,” in Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC), Oct. 2017, pp. 1001–1006, doi: 10.1109/SMC.2017.8122741.

[24] A. E. Mohamed, “Comparative study of four supervised machine learning techniques for classification,” Int. J. Appl. Sci. Technol., vol. 7, no. 2, pp. 5–18, 2017.

[25] S. Miao, H. Xu, Z. Han, and Y. Zhu, “Recognizing facial expressions using a shallow convolutional neural network,” IEEE Access, vol. 7, pp. 78000–78011, 2019.

[26] L. Zahara, P. Musa, E. Prasetyo Wibowo, I. Karim, and S. Bahri Musa, “The facial emotion recognition (FER-2013) dataset for prediction system of micro-expressions face using the convolutional neural network (CNN) algorithm based raspberry pi,” in Proc. 5th Int. Conf. Inform. Comput. Technol. (ICICT), Nov. 2020, pp. 1–9.

[27] K. Shiriha and M. Buddha, “Facial emotion detection using convolutional neural network,” Int. J. Sci. Eng. Res., vol. 11, no. 3, pp. 51–54, 2020.

MOHD NADHIR AB WAHAB (Member, IEEE) received the B.Eng. (Hons.) and M.Sc. degrees in mechatronics engineering from Universiti Malaysia Perlis, in 2010 and 2012, respectively, and the Ph.D. degree in robotics and automation system from the University of Salford, U.K., in 2017. He is currently a Lecturer with the School of Computer Sciences, Universiti Sains Malaysia. His main research interests include mobile robotics, computer vision, artificial intelligence, optimization, navigation, and path planning.

AMRIL NAZIR was formerly a Senior Research Scientist with the Malaysian Research and Development Institute. He is currently an Associate Professor with the College Technological Innovation, Zayed University, and the Consulting Director/Chief Architect at CODECOMPASS LLP. His research interests include artificial intelligence (AI), machine learning, data science, and big data.

ANTHONY TAN ZHEN REN received the B.Eng. degree (Hons.) in electrical and electronic engineering from the INTI International College Penang in partnership with the University of Bradford, U.K., in 2017, and the M.Sc. degree in embedded system engineering from Universiti Sains Malaysia, in 2021. He is currently working as a Product Development Engineer with Intel Corporation, Penang. His main research interests include artificial intelligence and computer vision.

MOHD HALIM MOHD NOOR (Member, IEEE) received the B.Eng. degree (Hons.) in 2004, the M.Sc. degree, in 2009, and the Ph.D. degree in computer systems engineering from the University of Auckland, New Zealand, in 2017. He is currently a Senior Lecturer with the School of Computer Sciences, Universiti Sains Malaysia. His research interests include machine learning, deep learning, computer vision, and pervasive computing.

MUHAMMAD FIRDAUS AKBAR (Member, IEEE) received the B.Sc. degree in communications engineering from International Islamic University Malaysia (IIUM), Malaysia, in 2010, and the M.Sc. and Ph.D. degrees from The University of Manchester, Manchester, U.K., in 2012 and 2018, respectively. From 2010 to 2011, he was a Research and Development Engineer with Motorola Solutions, Penang, Malaysia. From 2012 to 2014, he was an Electrical Engineer with Universiti Sains Malaysia (USM). His current research interests include electromagnetic, microwave nondestructive testing, microwave sensor, and imaging.

AHMAD SUFRIL AZLAN MOHAMED received the B.Eng. degree (Hons.) from Multimedia University, Malaysia, the M.Sc. degree from The University of Manchester, U.K., and the Ph.D. degree from the University of Salford, U.K. He is currently with the School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang, Malaysia. His research interests include image processing, video tracking, facial recognition, and medical imaging.