2023 6th International Conference on Information and Computer Technologies (ICICT) | 979-8-3503-0095-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICICT58900.2023.00022

Design Methodology for Single-Channel CNN-Based FER Systems

Dorfell Parra
Department of Electric and Electronic Engineering
Universidad Nacional
Bogotá D.C., Colombia
dlparrap@unal.edu.co

Carlos Camargo
Department of Electric and Electronic Engineering
Universidad Nacional
Bogotá D.C., Colombia
cicamargoba@unal.edu.co

***Abstract*—Facial Expression Recognition (FER) systems classify emotions by using geometrical approaches or Machine Learning (ML) algorithms such as Convolutional Neural Networks (CNNs). Due to their complexity, these FER systems need to be implemented on high-performance hardware, which makes them unsuitable for embedded devices. To address this challenge, we propose a methodology for the design of low-complexity, CNN-based FER systems. Our methodology includes data preprocessing, Local Binary Pattern (LBP) implementation, Data Augmentation (DA), and CNN design. Here, we also introduce the Model M6, a single-channel CNN that reaches an accuracy of 94% in less than 30 epochs. M6 has 306,182 parameters that correspond to 1.17 MB of memory. Therefore, our methodology and M6 model are feasible for implementation onto embedded systems capable of computing floating point operations. We validated our methodology and M6 model using 66 tests with 6 CNN models and 4 training parameters (batch size, learning rate, number of epochs, optimizer). This validation was performed using the Japanese Female Facial Expression (JAFFE) dataset and TensorFlow. In each test, the relationship between parameters, layers, overfitting, and underfitting was studied. Moreover, we present a step-by-step guideline on how to design the single-channel CNN and provide open-source code for readers interested in reproducing our work.**

Keywords—CNNs, FER, JAFFE, LBP, TensorFlow.

I. Introduction

Facial expression recognition (FER) systems are essential for applications such as human-computer communication, animation, psychiatry, and automobile safety. [1], [7], [8]. Machine Learning (ML) algorithms as Convolutional Neural Networks (CNN) have been shown to increase the performance of the FER systems [5]. However, due to the diversity in face phenotypes and variability in the structure of facial expressions from person to person, designing FER systems remains challenging. Additionally, the image acquisition process is subject to factors that increase the complexity of the classification problem, including poor illumination, face rotations, and noise (e.g., hair, glasses).

FER systems are typically designed using complex multi-channel ML architectures suitable for implementation with high-performance hardware, but not with embedded devices. Furthermore, there is a lack of guidelines and code examples in the literature to provide training to new FER system designers.

Here, we propose a methodology based on single-channel CNNs to design low-complexity FER systems for implementation in embedded devices. We also present a step-by-step guideline on how to implement FER systems based on CNNs and provide the source code for readers interested in reproducing our results. This manuscript is organized as follows. In Section II, we provide a background of previous work on FER systems. In Section III, we describe the image datasets used in this study, image preprocessing algorithms, and the proposed FER algorithm. The performance of our FER methodology and CNN model parameters that affect this performance are discussed in Section IV. We present the conclusions in Section V.

II. Related Work

Traditional FER systems detect facial prototypical expressions as happiness, sadness, anger, surprise, disgust, and fear. For instance, the Facial Action Coding System (FACS) proposed in 1977 by Erman and Friesen [1] linked these facial actions to expressions. In FACS, the classification parameters are associated with individual or group changes in the facial muscles, coded as Action Units (AUs). For example, the action of raising the inner brow, caused by the Frontalis and Pars Medialis muscles, is known as AU1; and the action of taking the "Tongue Out" is coded as AU19 [4]. Also, AUs can be additive when they appear independently and non-additive when different AUs modify each other. Traditional FER systems lack robustness due to the changes in AU structure from individual to individual, even for the same expression. Furthermore, the FER system's accuracy is affected by training datasets with poor illumination, low exposure, rotation of head and elevation of mandible [5].

CNNs have been proposed for the implementation of FER systems as they can extract feature maps from the input data and train their parameters [2], [5], [6], [7], and [8]. Usually, a CNN-based FER system is made up of the preprocessed input data and the classification architecture, which could have one or more channels aiming to increase the system's performance. Table I summarizes the architecture and accuracy of four FER systems we found in the literature.

979-8-3503-0095-6/23/$31.00 ©2023 IEEE
DOI 10.1109/ICICT58900.2023.00022

TABLE I. EXAMPLES OF THE ARCHITECTURE AND ACCURACY OF CNN-BASED FER SYSTEMS IN THE LITERATURE

Ref	Model	Accuracy	Architecture
[6]	I^2CNN for grayscale images	75.28%	Conv1: 3 × 3 × 96 MaxPooling1: 5 × 5 × 96 Conv2: 3 × 3 × 256 MaxPooling2: 3 × 3 × 256 Conv3: 3 × 3 × 384 Conv4: 3 × 3 × 384 Conv3: 3 × 3 × 256 MaxPooling3: 3 × 3 × 256 Dense: 4096 Dense: 4096 Dense: 7 SVM Classifier
[7]	WMDNN, channel 1: Partial VGG16 for grayscale images	90.86%	First VGG16 layers ending with: Conv5-1: 7 × 7 × 256 Conv5-2: 3 × 3 × 256 Conv5-3: 3 × 3 × 512 Pool5: 2 × 2 × 512 Flatten Dense: 500
[7]	WMDNN, channel 2: Shallow CNN for LBP images	88.33%	Conv1: 7 × 7 × 64 Subsampling: 2 × 2 × 64 Conv2: 3 × 3 × 256 Subsampling: 2 × 2 × 256 Dense: 500
[8]	Appearance feature based for LBP images.	89.33%	Conv1: 5 × 5 × 64 Pooling1: 2 × 2 × 64 Conv2: 5 × 5 × 128 Pooling2: 2 × 2 × 128 Conv3: 5 × 5 × 256 Pooling3: 2 × 2 × 256 Dense: 1024 Dense: 500 Dense: 6

In [6], the authors proposed a one-channel system called I^2CNN that uses five convolutional layers and one SVM and reaches an accuracy of 75% with the Japanese Female Facial Expression (JAFFE) dataset and 98.3% with the Cohn-Kanade (CK+) dataset. In both datasets, the CNN was trained and tested with six emotions.

In addition, [7] presented a two-channel approach named Weighted Mixture Deep Neural Networks (WMDNN) that integrates two CNNs: a partial Visual Geometry Group (VGG16) network and a shallow CNN. Its input images came from a Local Binary Pattern (LBP) of 72×72 pixels, which enhanced image textures. This two-channel approach achieved an accuracy of 92.21% with the JAFFE dataset and 97.02% with the CK+ images.

The FER system proposed by Kim et al. [8] integrates two CNNs, one using geometric features, and the other using appearance features from LBP images. This system achieves 91.27% with JAFFE and 96.46% with CK+.

Authors in [21] created a dataset using samples from multiple datasets (e.g., JAFFE, FER2013 and Cohn-Kanade). Then, they designed a CNN to classify positive and negative emotions. To synthesize images evoking positional emotions, a Generative Adversarial Network (GAN) was used. The work reports an accuracy of 80.38% and a loss of 0.4. However, the CNN model used in the above study included several Batch Normalization and Dropout layers that increased the model complexity and size.

Lastly, more extensive reviews of FER works recently published can be found in [22], and [23].

The aforementioned FER architectures entail several layers and channels that make them unfeasible to use in lightweight embedded systems. Additionally, the guidelines and open-source code to implement FER systems is lacking in the literature. Addressing these limitations will help us to advance the development of FER algorithms for embedded platforms.

III. PROPOSED METHODOLOGY

In this work, we propose a methodology to design a single-channel CNN-based FER system and share open-source code for those interested in replicating our results. In this Section, we describe the dataset, data preprocessing pipeline, Local Binary Pattern (LBP), Data Augmentation (DA) transformations, and Design Space Exploration (DSE) employed in this work.

A. Datasets

Datasets suitable for training FER systems are composed of several images that share characteristics as resolution and color. These datasets typically face limitations that include the following [1].

- Emotions are expressed at different intensities for each subject.
- There is a loss of authenticity when the subjects are aware that they are being photographed.
- Environmental conditions may limit or inhibit the subject's spontaneous expressions.

Among all the different datasets available, the most popular in the literature are the Cohn-Kanade dataset, proposed in 1998 [4] and extended as CK+ in 2010 [9], and the Japanese Female Facial Expression (JAFFE) dataset proposed in [10], [11].

The JAFFE dataset is composed of 213 images of 7 facial expressions (6 basic facial expressions + 1 neutral) with only posed expressions from 10 Japanese female models [10].

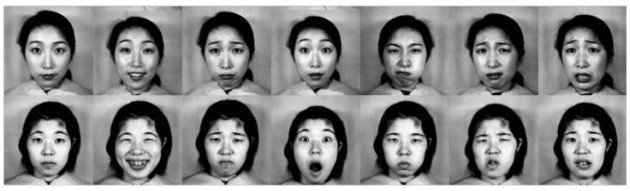


Fig. 1. Example images from JAFFE taken from [11].

The photographs were taken under strictly controlled conditions with similar lighting by using the acquisition setup presented [11]. Models have tied their hair away from their faces to avoid hiding their expressions. Fig. 1 shows 14 images from the JAFFE dataset.

B. Basic Preprocessing

The dataset preprocessing aims to improve the FER system's accuracy by eliminating irrelevant information such as background and adjusting parameters as the input size and resolution. Fig. 2 summarizes the steps involved in our dataset preprocessing pipeline.

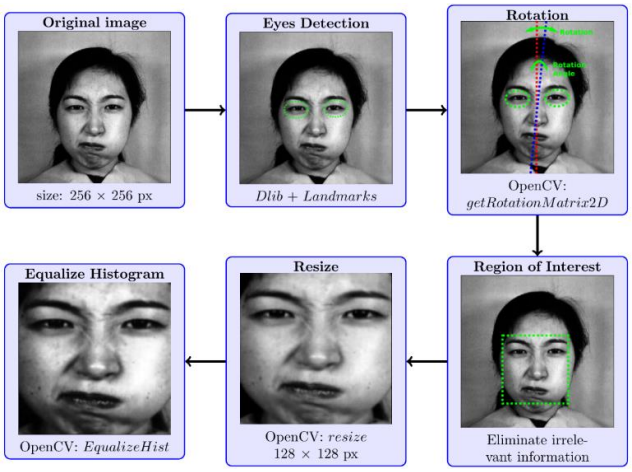


Fig. 2. JAFFE Preprocessing for FER. Adapted from [12].

The original images have a size of 256 x 256 pixels and came directly from the dataset. Faces are typically detected by using algorithms such as the Adaboost learning algorithm, proposed by Viola and Jones in 2004 [13]. In this work, the landmarks released by Davis King in [14] and first published in [15] are employed (see Fig. 3). These landmarks are read using a C++ toolkit of ML Algorithms known as Dlib [16].

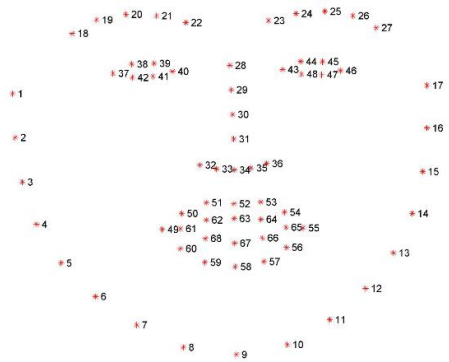


Fig. 3. Facial Point Annotations. Taken from [15].

Specifically, the methods used are *get_frontal_face_detector()* and *shape_predictor(landmarks)* following the algorithm proposed by Anas Khayata in [12]. For instance, the eyes detection algorithm uses the points 36-42 for the right eye and 42-48 for the left eye (Fig. 3). Next, the rotation angle is calculated by using a line between the eyes, while a vertical line traversing the image is used as a reference (Fig. 2, first row, third column). Rotation makes use of the functions *getRotationMatrix2D* and *warpAffine* from the OpenCV library [17]. Then, the Region of Interest (ROI) is cropped from the image, eliminating background and non-relevant parts such as the hair and the neck. The image is then resized to 128 × 128 pixels. Lastly, the image contrast is enhanced with Histogram Equalization using the OpenCV *equalizeHist()* function.

C. Local Binary Pattern (LBP)

The visual descriptor known as LBP is frequently used in FER systems [7], [8], due to the enhancement of textures associated with the AU.

To generate the LBP codes, a pixel from the grayscale image is first compared to the pixels around it in a 3 × 3 array. If the surrounding pixels are brighter than the center pixel, then the value one is assigned to that pixel. Otherwise, the center pixel becomes zero, as shown in Fig. 4.

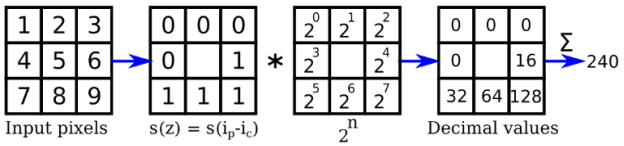


Fig. 4. LBP encoding example. Adapted from from [7], [8].

The following equations summarizes the encoding process

$$LBP = \sum_{n=1}^{N} s(i_p - i_c) \times 2^n,$$

$$s(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

where N is the number of neighboring pixels, and i_p and i_c represent the grayscale image values from the neighbors and the center. Once the binary values of the LBP code have been calculated, these values are converted to decimal values to obtain the LBP image. Implementation of LBP in this work was made by employing the Scikit-image library [18]. In particular, the function used is *local_binary_pattern* from the *skimage.feature* library. This function provides four methods for determining the pattern as described in [19]:

- **ror:** Rotation-invariant LBP.
- **uniform:** LBP with improved rotation invariance and uniform patterns.
- **nri_uniform**: LBP with uniform patterns but non rotation-invariant.
- **var:** LBP based on rotation invariant variance measures of the contrast of local image texture.

Fig. 5 shows the different LBP methods that were applied to the same JAFFE image. The LBP-var method was selected for the entire JAFFE dataset since it resulted in the smoothest images. Finally, using OpenCV.*resize* with LANCZOS4 as the interpolation method, the LBP image was resized to 128 × 128 pixels.

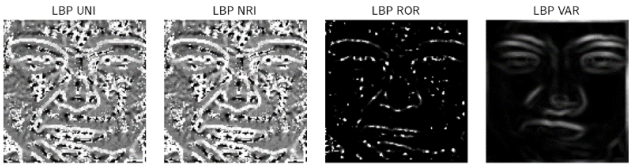


Fig. 5. LBP methods comparison with JAFFE.

D. Data Augmentation (DA)

Training CNNs requires a large number of images to get acceptable accuracy and avoid overfitting. As the JAFFE dataset size is 213 images, we increased the number of training samples using DA. In the DA approach, each training set sample is subject to transformations as rotation and flip in order to create new synthetic training data.

We employed Albumentation, a Python library for DA [20]. Fifteen transformations that include combinations of shift, scale, rotation, and flip were applied to images of six emotions [7], [8]. After this process, the number of training images increased from 213 to 2640. Fig. 6 shows some 16 DA examples from JAFFE resized to 64 × 64 pixels.

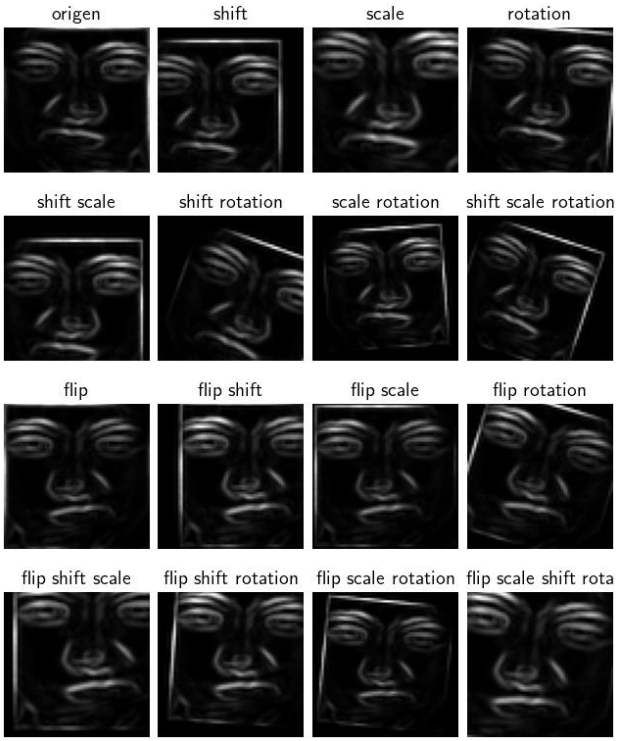


Fig. 6. Examples of DA with JAFFE.

E. Design Space Exploration (DSE)

The CNNs design is an iterative process that depends on knowledge about the input data and accuracy of the model. Moreover, expertise on CNNs will allow one to limit the DSE to tuning the most relevant parameters. Besides, some known tips are:

- Start with a convolutional layer with a few filters and increase the filters in the following layers. The number of filters should be multiples of two.
- Use a sub-sampling layer (e.g. *AveragePooling, MaxPooling)* after each convolutional layer to decrease the size of the feature maps.
- The number of neurons in the last layer (e.g. Dense) should be equal to the number of classes. Remember that for FER systems, each class corresponds to an emotion.
- Compare the performance of different optimizers (e.g. SGD, Adam) using metrics such as accuracy, loss and learning curves.
- Fine-tune the learning rate (a.k.a. lr or α) to improve the training performance. $\alpha = 0.001$ is a good starting point.
- Tune the batch size (bs) and the number of epochs to avoid underfitting and overfitting.

Then, the model's performance can be assessed using the learning curves for accuracy and loss. Hence, three scenarios are possible:

- **Underfitting:** The model doesn't reach a lower error value, and the training loss doesn't decrease no matter the number of epochs. Another scenario is that the training loss is decreasing but the training is halted, so it is needed to increase the number of epochs until the curve gets stable.
- **Overfitting:** The model learned the training dataset but failed to generalize; therefore, it presents high accuracy with the training dataset and low accuracy with the test dataset. The validation loss curve could decrease in the first epochs and then start to increase, creating a gap between it and the training loss.
- **Good fit:** the curves for the validation and training loss decrease to an acceptable value, and there is a small gap between them. Likewise, observe that increasing the number of epochs without control can result in overfitting.

IV. Experiment and Discussion

The experiments presented in this section aim to find a CNN-based model for FER systems with acceptable accuracy. Each test was designed to study the relationship between layers, parameters, and performance. To remove outliers, tests were repeated from 3 to 10 times, and the learning curves were used to estimate the convergence of each training. In addition, the source code and the training reports can be found in: https://gitlab.com/dorfell/fer_sys_dev/-/tree/master/00_sw

A. Experimental Setup

Experiments employed the preprocessed JAFFE dataset with 6 and 7 emotions. Although there are several algorithms that could be used to enhance the dataset (e.g., the Canny Edge Detector), the ones used in this work were taken from the literature. This includes the LBP with the "VAR" method and the DA with the Albumentation module [20]. Furthermore, the TensorFlow framework [3] was utilized for training on Ubuntu

20.04. The laptop used was a Legion 5 with an AMD Ryzen 7 4800h processor, RAM of 16 GB, and a NVIDIA GTX 1660 Ti GPU.

B. Performance Analysis

Table II presents the average accuracy and loss of some model architectures tested. The parameters under observation were the batch size, the learning rate, the optimizer's performance (e.g., SGD, Adam) and the *Dropout* effect.

TABLE II. EXAMPLES OF TESTED ARCHITECTURE.

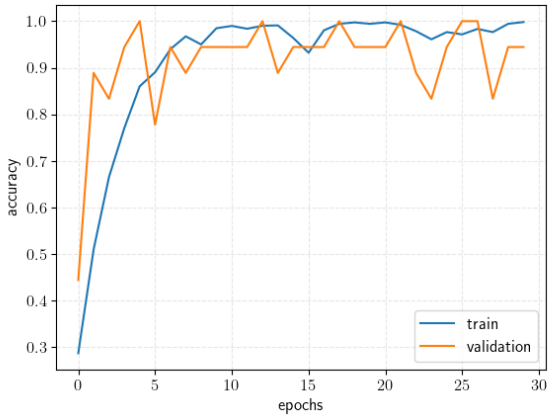
Model	Architecture	Parameters	Accuracy	Loss
M3	Input: 128 × 128 Conv1: 5 × 5 × 64 MaxPooling: 2 × 2 Conv2: 5 × 5 × 128 AveragePooling: 2 × 2 Conv3: 5 × 5 × 256 AveragePooling: 2 × 2 Dense: 1024 Dense: 512 Dense: 7	α = 0.001, Batch size = 64, Opt = Adam, Epochs = 30.	75.67%	1.02
M4	Input: 64 × 64 Conv1: 5 × 5 × 64 MaxPooling: 2 × 2 Conv2: 5 × 5 × 128 MaxPooling: 2 × 2 Conv3: 5 × 5 × 256 MaxPooling: 2 × 2 Dense: 7	α = 0.0001, Batch size = 64, Opt = Adam, Epochs=30.	69.05%	1.26
M6	Input: 64 × 64 Conv1: 5 × 5 × 32 MaxPooling: 2 × 2 Conv2: 5 × 5 × 64 MaxPooling: 2 × 2 Conv3: 5 × 5 × 128 MaxPooling: 2 × 2 Dense: 6	α = 0.001, Batch size = 32, Opt = Adam, Epochs=30.	94.44%	0.31

Model M3 used input images of 128 × 128 and 5 × 5 kernels. The learning curves obtained after 30 epochs showed that the model had an accuracy of 75% but tended to overfit.

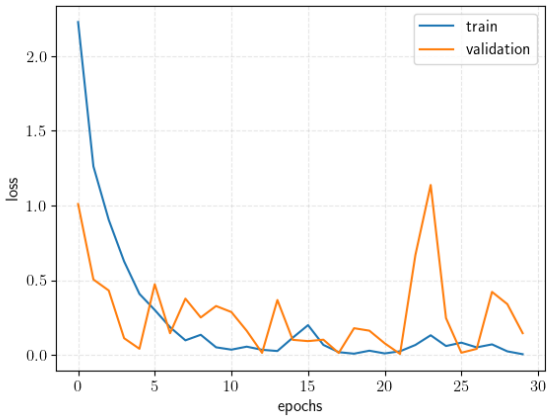
Model M4 was based on Model M3 but included only 1 *Dense* layer to avoid overfitting. Furthermore, the input size changed to 64 × 64 and the accuracy curve remained steady after 15 epochs. The learning curves showed a validation accuracy of around 70% but the gap between the training and the validation loss indicated that overfitting was still present.

For Model M6 the number of filters in the *Convolutional* layers was reduced to 32, 64, and 128. In addition, the *Dense* layer includes 6 neurons instead of 7, thus training was made with the dataset including 6 emotions, as found in the literature. As can be seen in the learning curves show in Fig. 7, the validation accuracy is stabilizing around 94% and the validation loss around 0.4. Furthermore, the gap between the training and validation loss is approximately 0.4, which indicates an acceptable fit.

Figure 8 shows how the model's performance is affected when changing parameters such as the learning rate, type of sub-sampling layer, and dropout. We observed that having too small or too large learning rates could lead the optimizer to miss the optimal weights. Besides, as model M6 has a small number of trainable parameters, adding a dropout layer will decrease the model's accuracy.



a)



(b)

Fig. 7. Learning curves for model M6. In (a) accuracy and in (b) loss.

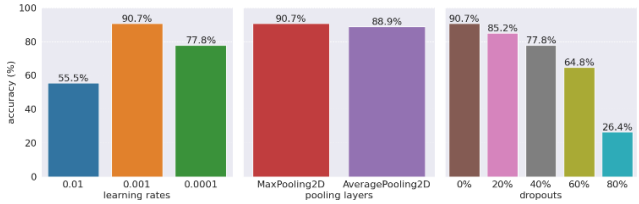


Fig. 8. Effects of the learning rate, the subsampling layer, and the dropout on the M6 model's performance.

Moreover, the number of parameters in M6 is 306,182 and the model size is 1.17 MB. Which could be feasible for custom hardware-software architectures supporting floating operations.

Lastly, Figure 9 shows the accuracy and number of trainable parameters for M6 and the models from the related work section.

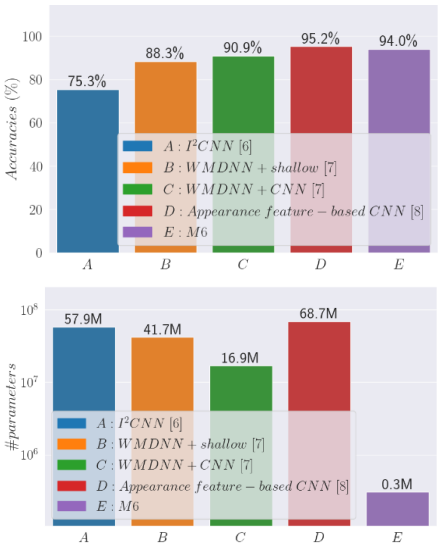


Fig. 9. Accuracies (top) and number of trainable parameters (bottom) for M6 and the models from the related work section.

V. Conclusions

This work faces the problem of designing CNN-based models for FER that fit on devices with limited resources. The methodology proposed describes the steps needed to enhance image quality (e.g., data preprocessing, LBP and the DA). Besides, the relationships between training parameters, layers, and model performance are studied.

Moreover, a single-channel CNN-based FER system (a.k.a. M6) that achieves an average accuracy of 94% is introduced. This model has been trained with TensorFlow and the preprocessed JAFFE dataset.

In the future, we would like to test the model with other data sets available and assess if it could be used with real-world images.

Acknowledgment

The authors would like to acknowledge Professor David Escobar (Department of Biomedical Engineering, Lerner Research Institute, Cleveland Clinic) for reviewing the paper and giving significant feedback.

References

[1] V. Bettadapura, “Face expression recognition and analysis: The state of the art,” CoRR, vol. abs/1203.6722, pp. 1–27, 2012.

[2] M. Z. Uddin, W. Khaksar, and J. Torresen, “Facial expression recognition using salient features and convolutional neural network,” IEEE Access, vol. 5, pp. 26 146–26 161, 2017.

[3] “TensorFlow: An open-source software library for machine intelligence,” 2022. [Online]. Available https://www.tensorflow.org/.

[4] Y. Tian, T. Kanade, and J. Cohn, “Recognizing action units for facial expression analysis,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 97–115, 2001.

[5] Xie, Siyue, and H. Hu, “Facial expression recognition using hierarchical features with deep comprehensive multipatches aggregation convolutional neural networks,” IEEE Transactions on Multimedia, vol. 21, no. 1, pp. 211–220, 2019.

[6] C. Zhang, P. Wang, K. Chen, and J. Kamarainen, “Identity-aware convolutional neural networks for facial expression recognition,” Journal of Systems Engineering and Electronics, vol. 28, no. 4, pp. 784–792, 2017.

[7] 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, 2018.

[8] J. Kim, B. Kim, P. Roy, and D. Jeong, “Efficient facial expression recognition algorithm based on hierarchical deep neural network structure,” IEEE Access, vol. 7, pp. 41 273–41 285, 2019.

[9] 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,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, pp. 94–101, 2010.

[10] M. Lyons, M. Kamachi, and J. Gyoba, “The japanese female facial expression (jaffe) dataset,” 1998. [Online]. Available https://doi.org/10.5281/zenodo.3451524.

[11] M. Lyons, M. Kamachi, and Gyoba, “Coding facial expressions with gabor wavelets (ivc special issue),” Modified version of a conference article, that was invited for publication in a special issue of Image and Vision Computing dedicated to a selection of articles from the IEEE Face and Gesture 1998 conference. The special issue never materialized., 2020. [Online]. Available: https://zenodo.org/record/4029680

[12] K. Anas, “Facial expression recognition in jaffe database.” [Online]. Available: https://github.com/anas-899/facial-expression-recognition-Jaffe

[13] V. Paul and J. Michael J., “Robus real-time object detection,” International Journal of Computer Vision, vol. 57, no. 2, pp. 137–154, December 2004.

[14] K. David, “Dlib-models.” [Online]. Available: https://github.com/davisking/dlib-models/

[15] S. Christos, T. Georgios, Z. Stefanos, and P. Maja, “300 faces in-the-wild challenge: The first facial landmark localization challenge,” Proceedings of IEEE Int’l Conf. on Computer Vision (ICCV-W), 2013. [Online]. Available: https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

[16] K. Davis E., “Dlib-ml: A machine learning toolkit,” Journal of Machine Learning Research, vol. 10, no. 2, pp. 1755–1758, 2009.

[17] Itseez, “Open source computer vision library.” [Online]. Available: https://github.com/itseez/opencv

[18] S. van der Walt, J. L. Schonberger, J. , Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors, “scikit-image: image processing in Python,” PeerJ, vol. 2, p. e453, 6 2014. [Online]. Available: https://doi.org/10.7717/peerj.453

[19] skimage, “local binary pattern.” [Online] Available: https://scikit-image.org/docs/stable/api/skimage.feature.html?highlight=local binary pattern#skimage.feature.local binary pattern

[20] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, “Albumentations: Fast and flexible image augmentations,” Information, vol. 11, no. 2, 6 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[21] H. Burrows, J. Zarrin, L. Babu-Saheer, and M. Maktab-Dar-Oghaz, “Realtime Emotional Reflective User Interface Based on Deep Convolutional Neural Networks and Generative Adversarial Networks,” Electronics, vol. 11, no. 1, p. 118, Dec. 2021, doi: 10.3390/electronics11010118.

[22] Anand M, Dr. S. Babu, "A Comprehensive Investigation on Emotional Detection in Deep Learning,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 8, Issue 1, pp.115-122, January-February-2022. Available at doi : https://doi.org/10.32628/CSEIT228111

[23] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," in *IEEE Transactions on Affective Computing*, vol. 13, no. 3, pp. 1195-1215, 1 July-Sept. 2022, doi: 10.1109/TAFFC.2020.2981446.