Learnable pooling weights for facial expression recognition

M. Amine Mahmoudi, Aladine Chetouani, Fatma Boufera, Hedi Tabia

Learnable pooling weights for facial expression recognition

M. Amine Mahmoudi[a], Aladine Chetouani[b], Fatma Boufera[a], Hedi Tabia[c]

aMustapha Stambouli University of Mascara, Algeria

bPRISME laboratory, University of Orleans, Orleans, France

cIBISC, Univ Evry, Université Paris-Saclay, Evry, France

Pooling layers are spatial down-sampling layers used in convolutional neural networks (CNN) to gradually downscale the feature map, increase the receptive field size and reduce the number of the parameters in the model. The use of pooling layers leads to less computing complexity and memory consumption reduction but also introduces invariance to certain filter distortions which may induce subtle detail loss. This behaviour is undesired for some fine-grained recognition tasks such as facial expression recognition (FER) which highly relies on specific regional distortion detection. In this paper, we introduce a more filter distortion aware pooling layer based on kernel functions. The proposed pooling reduces the feature map dimensions while keeping track of the majority of the information fed to the next layer instead of ignoring part of them. The experiments on RAF, FER2013 and ExpW databases demonstrate the benefits of such layer and show that our model achieves competitive results with respect to the state-of-the-art approaches.

1. Introduction

Facial expression recognition (FER) research aims at classifying the human emotions given facial images as one of seven basic emotions: happiness, sadness, fear, disgust, anger, surprise and neutral. FER finds applications in different fields including security, intelligent human-computer interaction, and clinical medicine. Recently, many FER works [1,15,19,23,32–34] based on Convolutional Neural Networks (CNNs) have been proposed. Although these works mainly differ by the model architectures and the used databases, convolution and pooling layers are often employed.

Pooling layers generally enhance the network performance and are mainly used for the gradual spatial down-sampling of the feature map. This will reduce the parameters number and thus reduces the memory consumption and computing complexity. In addition, pooling layers increase the receptive field size of the intermediate neurons allowing the latter to receive information from a larger area of the image. However, pooling layers introduce invariance to slight distortion which decreases the network performance on FER tasks as this distortion may cause the loss of some discriminative details [10].

In the literature, three conventional pooling methods have usually been employed with CNNs, namely: (1) max pooling, (2) average pooling and (3) strided convolution, having each their advantages and drawbacks. Max pooling, for instance, only keeps the largest input values assuming that the rest of values are not representative and do not bring relevant information. This assumption however is not always true, especially in the last layers of the network where even the small values represent a very relevant information. Therefore max pooling dramatically reduces the amount of useful information in the forward pass. Moreover, max pooling wrongly affects the learning of the network in the backward pass, since only one branch is activated in each input neighborhood. In an average pooling layer, all the inputs equally contribute to the output computation. This causes a constant and gradual attenuation of the contribution of individual neurons in the backward and forward passes [25]. Strided convolution is simply a convolution layer with a stride bigger than one. This kind of layer is used in some very deep networks like ResNet [5], whereas max and average pooling are used in mid-size networks like VGG [26] and GoogleNet [27].

Although, these pooling methods are easy to compute from an input neighborhood, they omit important discriminant details which are crucial to many fine-grained classification problems. Particularly for FER, in which, we are more interested in detecting specific distortions of facial regions rather than simply identifying it in a given location (which is the case in a max-pooling operation [2]). To handle this problem, we introduce in this paper, a more filter-distortion aware pooling layer based on a kernel function which reduces the feature map dimensions while keeping

* Corresponding author.
E-mail address: mohamed.mahmoudi@univ-mascara.dz (M.A. Mahmoudi).

track of the most discriminant information for FER instead of ignoring part of it. We show that the proposed layer improves the model performance in FER task, without many additional parameters.

Our contributions

We propose a FER method based on a CNN model to which we specifically designed a novel pooling layer that retains the down-sampling advantage of the ordinary pooling function and brings several new features. The proposed pooling layer has learnable weights which generalize standard pooling functions (i.e. max and average pooling) and, additionally encodes patch-wise non-linearity which in turn improves the discrimination power of the full network. The novel pooling is completely differentiable and can be used at any level of the network, allowing an end-to-end learning.

The remainder of this paper is organized as follow: Section 2 reviews similar works that have been proposed for FER using deep learning techniques, pooling based networks. Section 3 introduces the proposed pooling layer for FER. Section 4 presents the different conducted experiments and their related results. Section 5 concludes the paper.

2. Related work

Recently many works have been proposed for FER using deep learning techniques. The majority of these works focuses on the CNN architecture and its tuning on either laboratory controlled or wild images. An extensive survey has been proposed by Li and Deng [17] for more details. All presented CNN based methods [17] use one or many conventional pooling layers (max/average pooling, or strided convolution). As stated in Section 1 these pooling layers introduce invariance to slight distortion which may decrease the network performance on fine-grained classification tasks.

Several pooling methods have been proposed to overcome this limitation and thereby improves the performance of CNNs. In [22] a bilinear pooling method for fine-grained recognition was proposed. Inspired from the second order pooling model introduced by Tenenbaum and Freeman [29], it consists of using two CNNs as feature extractors and combine their outputs by multiplying each location of the resulting feature maps using the outer product. The result is then sum-pooled to obtain the final image descriptor. This model has been improved later in [21] by additionally applying a matrix function normalization. Two matrix functions have been used for this purpose namely the matrix logarithm and the matrix square-root. However, these models are high dimensional and they could be impractical for a multitude of image analysis. In [9] two compact bilinear representations of these models have been proposed with the same discriminative power as the full bilinear representation, yet with only a few thousand dimensions. The later was further generalized in [6] in the form of Taylor series kernel. The proposed method captures high order and non-linear feature interactions via compact explicit feature mapping. The approximated representation is fully differentiable, thus the kernel composition can be learned together with a CNN in an end-to-end manner. In [1] the authors explored the benefits of using a manifold network structure for covariance pooling of second-order statistics for both videos and images in the context of FER.

All these methods are always plugged at the end of the network, right between the convolution layers and the fully connected layers. They act as a basis expansion layers, increasing thereby the discrimination power of the fully connected layers. This discrimination power is back-propagated through the convolution layers allowing the network to learn in an end-to-end fashion. These methods have attracted increasing attentions, achieving better performance than classical, first-order networks in a variety of computer vision tasks. Even-thought these methods increase the CNN performance, they are enable to learn by themselves and rely entirely on CNN architecture. Furthermore, how to effectively introduce higher-order representation in earlier layers for improving non-linear capability of CNNs is still an open problem.

Recently, Gao et al. [11] addressed this problem and a novel network model has been proposed introducing global second-order pooling across from lower to higher layers for exploiting holistic image information throughout a network. Wang et al. [30] proposed to replace the convolution layers in a CNN by kernel based layers called kervolutions. The use of these layers increases the model capacity to capture higher order features at the convolutional phase. For the same purpose, Hyuan et al. [14] proposed a new pooling method called universal pooling. The method intends to generate pooling function which better fits any problem given a dataset. Universal pooling has been inspired by attention methods and can be considered as a channel-wise form of local spatial attention. The strength of these methods relies on the fact that they capture additional discriminant information compared to conventional pooling techniques. This makes them more suitable for fine-grained classification problem. In this paper, we build upon these works and introduce a novel pooling layer that not only uses all input information but also extracts linear and non-linear relations between features. To do so, we leverage kernel functions which allow to generalise linear pooling while capturing higher order information.

3. Proposed method

We propose an end-to-end model to perform the FER task. Our network architecture is simple. It is designed to capture discriminant facial features through successive layers of multiple non-linear transformations and representations. It follows standard CNN as it alternates convolutional and pooling layers and ends with a fully connected softmax activation layer (see Fig. 1). The convolutional layers learn several filter weights which are convolved with the input facial image and produce a set of feature maps. The filter weights are learned such that the final classification score is high (categorical cross entropy loss is employed, see Section 3.2).

The novelty in this work is specific pooling layers which are more sensitive to subtle details in feature maps than standard pooling techniques (i.e. max-pooling, average-pooling, etc). This is ensured by adding learnable weights to the pooling layers, similarly as in convolutional layers, while reducing the feature map size. By doing so, the standard pooling techniques can be seen as a particular case of our new pooling with fixed (non-learnable) weights. In Section 3.1, we present our proposed pooling layer used for boosting the FER task.

3.1. Learnable pooling

The proposed pooling layer is similar to an ordinary pooling one in the way that it applies a pooling function on a specific location and a specific stride. The difference from previous poolings corresponds to the capability to dynamically extract more relevant features from the input map. This is particularly performed by learning different pooling weights for each feature map. These weights are learned in a similar fashion as convolutional weights but with a single depth output (see Fig. 1). Finally, a combination of the original feature map and the resulting weights is computed using a specific function. This function is carefully chosen to capture linear and non linear relations between both the weights and

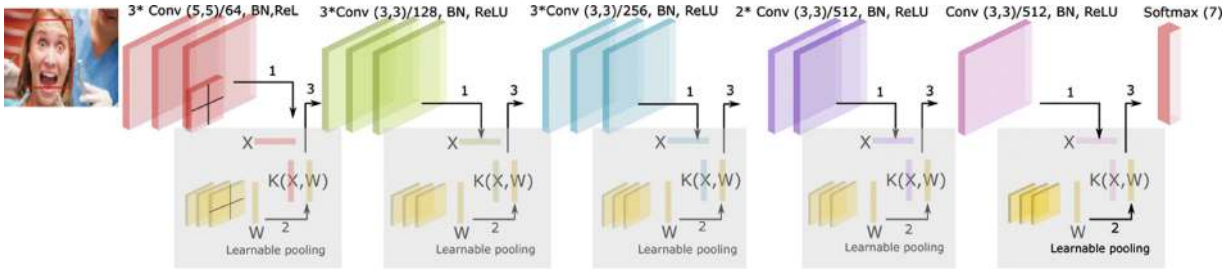

Fig. 1. Our proposed network architecture for FER task. The CNN alternates convolutional layers and specifically designed layers. It ends by a fully softmax activation layer. Each convolutional layer is followed by batch normalization and rectified linear unit activation.

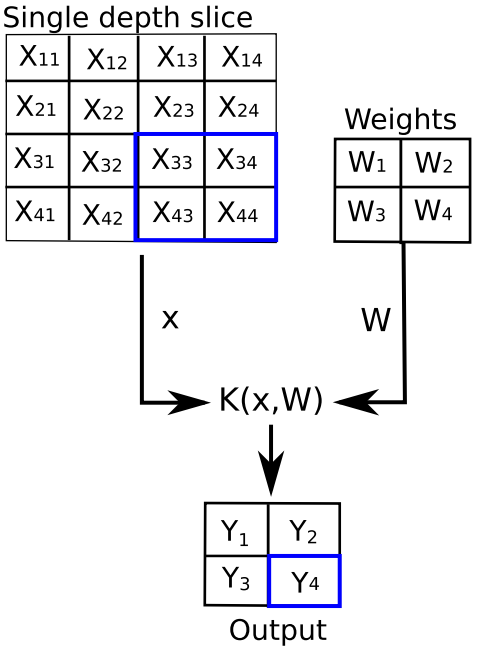

Fig. 2. The processing of our pooling layer is the same as the standard pooling layer in the way that it reduces the spatial dimensions of a given feature map. Our pooling layer uses learnable weights to encode relevant relations between features via a kernel function.

the original feature map. The output of our pooling layer is a new feature map with reduced high and width.

Formally and as shown in Fig. 2, we consider a flattened feature map vector $x = \{x_1, x_2, \ldots, x_d\}$ and a vector of pooling weights $w = \{w_1, w_2, \ldots, w_d\}$. In Fig. 2, d is equal to 4 which corresponds to a 2×2 pixels. The vector w can be seen as a second feature map which is dynamically learned. In order to capture linear and non-linear relations between x and w, we employ a Symmetric Positive Definite (SPD) function $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The choice of the SPD function is motivated by the fact that standard pooling (e.g. average pooling) can be considered as a linear combination of fixed filter weights and the feature map values in a particular location. For instance given the feature map vector x and a set of non-learnable weights $w = \{1/d, 1/d, \ldots, 1/d\}$, the average pooling can be computed as: $\sum_{i=1}^{d} w_i * x_i = x^T w$ which is the dot product of both vectors x and w, and corresponds to the inner product in $\mathbb{R}^d$. By employing a SPD function, we emulate an inner product in a higher dimensional space after a non linear mapping of both x and w vectors. Thus, the pooling operation turns out to be

$$\langle \varphi(x), \varphi(w) \rangle \approx \mathcal{K}(x, w). \tag{1}$$

In this work, we employed three different functions defined on the feature map space which has an Euclidean structure $\mathbb{R}^d$.

- Linear kernel:

$$\mathcal{K}(x, w) = x^T w, \quad x, w \in \mathbb{R}^d. \tag{2}$$

- Polynomial kernel:

$$\mathcal{K}(x, w) = (x^T w + r)^n, \quad x, w \in \mathbb{R}^d, r \geq 0. \tag{3}$$

- Gaussian kernel (RBF Kernel):

$$\mathcal{K}(x, w) = e^{-\frac{\|x-w\|^2}{2\sigma^2}}, \quad x, w \in \mathbb{R}^d, \sigma > 0. \tag{4}$$

The linear kernel (Eq. (2)) looks at the similarity between the feature map vector x and the filter weight vector w. Starting form $n > 1$ in Eq. (3), the polynomial kernel encodes not only the linear relation between both x and w vectors, but also non-linear relations between them. Thanks to the exponential term in Eq. (4), the Gaussian kernel expands the pooling non-linearity to the infinity. This expansion can also be reached by other functions, such as the Laplacian kernel defined by $\mathcal{K}(x, w) = e^{-\alpha\|x-w\|}$, or the Abel kernel defined as $\mathcal{K}(x, w) = e^{-\alpha|x-w|}$, where, $\alpha > 0$ in both kernels.

The proposed pooling preserves the main purpose of a standard pooling layer which corresponds to the down-sampling of the input feature map. But it not only summarizes the presence of specific features in patches, it also captures the non-linear relations between these features.

3.2. Learning

Our network ends with a fully connected layer to make sure that all activations in the previous layer are connected to the last layer and to allow the pooled 2D feature maps to be converted into a vector of probabilities for FER. In this work, we chose to use the traditional softmax layer, with a cross entropy loss, to simply force features of different expressions to remain apart. Many authors have proposed advanced losses for FER such as the center loss [31], the island loss [4], and the locality-preserving loss [8]. However here we opt for a simple softmax layer and a cross entropy loss to demonstrate the efficiency of the proposed pooling layer.

Given a facial image I with a label vector y of y_i elements ($y_i = 1$ if I belongs to C_i otherwise $y_i = 0$ where C_i indicates the ground truth expression of the face in I), the objective of our learning problem is to minimize the cross entropy loss over the set of C classes :

$$CE = \sum_{i}^{C} y_i \log(f(I)_i), \tag{5}$$

where $f(I)_i$ stands for the softmax activation of the ith class.

The learnable pooling weights are initialized using He normal function. It draws samples from a truncated normal distribution centered on 0 with a standard deviation given by:

$$stddev = \sqrt{\frac{2}{N}} \tag{6}$$

Where N is the number of input units in the weight tensor.

4. Experiments

In order to evaluate our method, several experiments have been conducted on three well-known datasets, namely the RAF-DB [19], ExpW [35] and FER2013 [12].

- The RAF-DB Li et al. [19] stands for the Real-world Affective Face DataBase. It is a real-world dataset that contains 29,672 highly diverse facial images, downloaded from the Internet. With manually crowd-sourced annotation and reliable estimation, seven basic and eleven compound emotion labels are provided for the samples. This dataset is divided in training and validation subsets.
- The ExpW Zhang et al. [34] stands for the EXPression in-the-Wild dataset. It contains 91,793 facial images, faces downloaded using Google image search. Each of the face images was manually annotated as one of the seven basic expression categories.
- The FER2013 database was first introduced during the ICML 2013 Challenges in Representation Learning Goodfellow et al. [12]. This database contains 28,709 training images, 3589 validation images and 3589 test images, with seven expression labels: fear, happiness, anger, disgust, surprise, sadness and neutral.

4.1. Training process

The only preprocessing which we have employed on all experiments is cropping the face region and resizing the resulting images to 100×100 pixels.

In order to demonstrate the efficiency of the proposed pooling, we preferred to build a simple CNN from scratch rather than using a pre-trained one. We have used Adam optimiser with a learning rate varying from 0.001 to 5e-5. This learning rate is decreased by a factor of 0.63 if the validation accuracy does not increase over ten epochs. To avoid over-fitting we have also augmented the data using a range degree for random rotations of 20, a shear intensity of 0.2, a range for random zoom of 0.2 and randomly flip inputs horizontally.

As shown in Fig. 1 our model architecture is quite simple and can effectively run on cost-effective GPUs. It is composed of five convolutional blocks. Each block consists of a convolution, batch normalization and rectified linear unit activation layers. The use of batch normalization [36] before the activation brings more stability to parameter initialization and achieves higher learning rate. Each of the five convolutional blocks is followed by the proposed pooling layer and a dropout layer. In the following we refer to this network architecture as (Model-1).

4.2. Ablation study

This section explores the impact of the use of the proposed learnable pooling layer on the overall accuracy of a FER CNN. We evaluated the performance of the same network architecture Model-1 but with different pooling techniques. First, we used Model-1 with standard max and average poolings. After that, we replaced these poolings by our learnable layers. We studied the behaviour of four different kernel functions, namely; (1) the linear kernel, (2) the second-order polynomial kernel, (3) the third-order polynomial kernel and (4) the Gaussian RBF kernel.

The experiments are conducted with the same training parameters as described above. Table 1 presents the results of our FER model using standard pooling with comparison to the proposed learnable pooling. From this table, one can notice that considering Model-1 architecture, the use of max or average pooling gives approximately the same results with a slight improvement when max pooling is employed. In the case of max pooling, Model-1 attains 75.91%, 87.05% and 70.49% of accuracy rate on respectively ExpW, RAF-DB and FER2013 datasets. When using average pooling layers instead of max pooling, Model-1 reaches an accuracy rate of 75.74%, 86.89%, and 70.13% for respectively ExpW, RAF-DB and FER2013 datasets. However in contrast to these two cases, the use of learnable pooling layers in Model-1 architecture considerably increases its accuracy. As reported in Table 1, the accuracy of Model-1 in which we use a learnable pooling with a linear kernel increases the accuracy up to 0.4% for ExpW, 3.75% for RAF-DB and 0.2% for FER2013. Model-1 using learnable pooling layers with linear kernels performs at least as good as the use of layers with max or average pooling. This behaviour can be explained by the fact that the linear kernel can automatically learn the suitable pooling method from a continuum of methods which include the average and the max pooling as particular cases. Although our proposed pooling layer increases the number of learnable parameters, compared to the max and the average pooling, the use of the simple linear kernel gives more flexibility to the pooling since it acts as a decision maker of which weights to fix or use for each particular filter.

Table 1
Accuracy rate of our proposed approach for different pooling strategies. In this table, Model-1 architecture is used with the indicated pooling method.

Pooling	ExpW	RAF-DB	FER2013
Max	75.91%	87.05%	70.49%
AVG	75.74%	86.89%	70.13%
Linear kernel	76.28%	90.81%	70.69%
2nd-order Poly	76.64%	92.87%	70.88%
3rd-order Poly	76.81%	93.21%	71.35%
Gaussian RBF	76.42%	92.74%	70.74%

Following the same principle, we further studies the impact of the usage of three additional kernels; (1) the second-order polynomial kernel with $r = 0$, (2) the third-order polynomial kernel with $r = 0$ and (3) the Gaussian kernel with $\gamma = 0.9$. Although the computational complexity of these kernels is higher compared to the max and the average pooling, they strongly improve the model accuracy when used. As shown in Table 1, the same model architecture but using the third-order polynomial kernel outperforms the other methods. The use of this kernel improves the accuracy rate close to 1% for most of the databases. Less efficient than the third-order polynomial kernel but also computationally expensive is the Gaussian kernel. It increases the accuracy up to 0.5% for ExpW and FER2013 comparing to max and average poolings and 5% for RAF-DB. Moreover, the Gaussian kernel takes a considerable time to converge. Finally, according to Table 1, the second-order polynomial kernel is less efficient than the third-order one but remains better than the Gaussian kernel. It merely outperforms the linear kernel with slightly higher complexity. However, it is the fastest kernel to converge compared to the third-order polynomial and the Gaussian kernels.

To further compare the proposed pooling technique with standard ones, we display in Fig. 3 the output image after each pooling layer. We compared max and average pooling with the third order polynomial pooling. As depicted in Fig. 3, our pooling method is able to capture more details than the standard pooling techniques. The visualizations show that the third order pooling captures relevant features which likely correspond to the expression action units e.g. the outlines of the mouth and the eyes. Moreover, the third order pooling outperforms the other techniques in discarding non-informative regions. One can clearly notice particularly in the two last layers, even when the results are abstract and difficult to interpret, that the learnable pooling keep activation of well localized features (nose, mouth and eyes). On the contrary, one can also notice some common activated regions particularly in the ear-

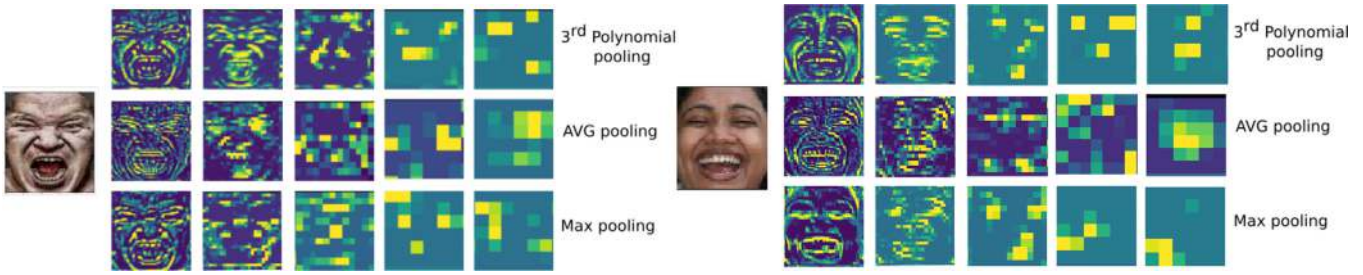

Fig. 3. Visualisation of the outputs from the pooling layers. These visualisations are generated from two facial expressions (the face in the left, and the face in the middle). Given an input image, we show the feature maps after each of the five pooling layers used in our CNN. The first row shows the feature maps after third polynomial kernel based pooling. The second and the third rows present feature maps after the standard average and max pooling respectively.

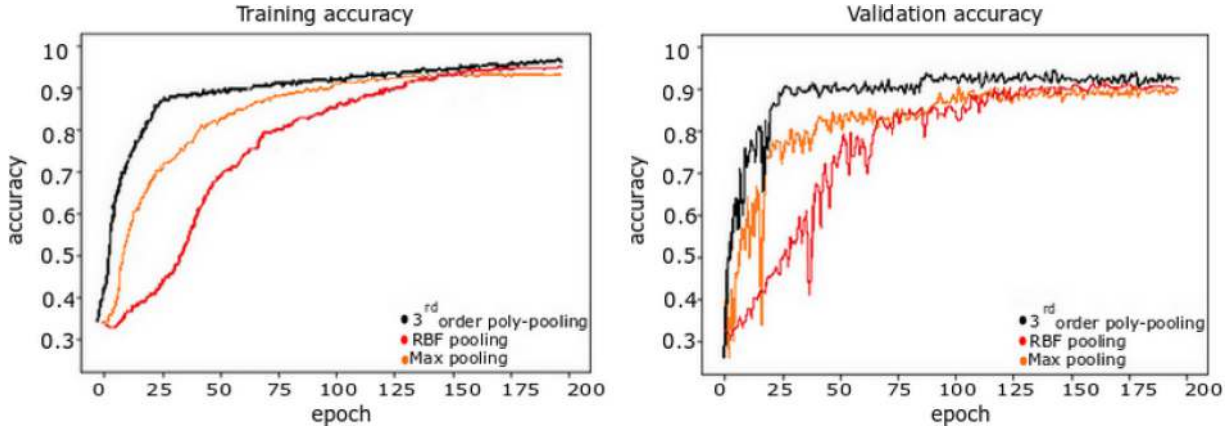

Fig. 4. Visualizations of accuracy versus epoch plots. This figure reports the impact of the learnable pooling on the convergence speed of the used CNN. A comparison between the performance of the CNN using max pooling, the third order polynomial, and the RBF pooling methods on the RAF-DB dataset.

lier layers, this can be explained by the fact that our pooling encompasses standard poolings thanks the linear term in the polynomial kernel. In Fig. 4, we report the training accuracy and validation accuracy versus epoch plots of our CNN when using the max pooling, the RBF, and the third order kernel pooling layers. These plots give an idea about the influence of the different pooling techniques on the convergence rate of the network. The sub-Figures demonstrate that the third order based pooling has an important impact on the convergence speed of the network compared to the max and the RBF based poolings. Although the computational complexity of the third order kernel is higher than the max pooling, it is still able to converge to a higher validation accuracy in less epochs.

Note that in the literature, few researchers claim that the standard max pooling performs a noise removal arguing that it gets rid of noisy features and also brings denoising along with dimensionality reduction [10]. On the contrary, the average pooling only carries out dimensionality reduction. Thus, the max pooling is generally considered to achieve better performance than average pooling. However, by using our proposed pooling, we demonstrate that the majority of the features in the input feature map are relevant. By leveraging kernel functions, non-linear relation between features in a given patch are captured and this allows to performs better than standard max and average pooling.

These results demonstrate that non-linear relations between features in the feature map produced after a convolutional layer are beneficial for the overall accuracy of the FER problem. The use of the third order pooling kernel allows to achieve the best performance compared to standard pooling techniques as well as the different studied kernels.

Table 2
Accuracy rate of our proposed approach and state of the art approach.

Methods	ExpW	RAF-DB	FER2013
Linear kernel	76.28%	90.81%	70.69%
2nd-order Poly	76.64%	92.87%	70.88%
3rd-order Poly	76.81%	93.21%	71.35%
Gaussian RBF	76.42%	92.74%	70.74%
Tang et al. [28]	–	–	71.16%
Guo et al. [13]	–	–	71.33%
Kim et al. [15]	–	–	73.73%
Bishay et al. [3]	73.1%	–	–
Lian et al. [20]	71.9%	–	–
Acharya et al. [1]	–	87%	–
Kuo et al. [16]	–	65.52%	–
Deng et al. [7]	–	68.2%	–
Li et al. [18]	–	74.2%	–
Liu et al.[24]	–	73.19%	–

4.3. Comparison with the state-of-the-art

In this section, we compare the performance of our FER model which uses the proposed learnable pooling with respect to several state-of-the-art methods. The obtained results are reported in Table 2. According to Table 2, our proposed model outperforms the state-of-the-art methods on the ExpW dataset. The best accuracy rate is 76.81% and has been reached using the third order polynomial kernel. The second order polynomial kernel gives 76.64% while the linear kernel achieves 76.28% as accuracy rate. Finally, using the Gaussian kernel our model achieves 76.42% of accuracy.

On RAF-DB dataset, the accuracy of our model is also superior to state-of-the-art methods. Using the third order polynomial pooling, our model accuracy exceeds the other methods by more than 1% and it obtains 93.21%. One can also notice that all kernels outperform state-of-the-art methods. Using a linear kernel, our model achieves 90.81% of accuracy while 92.87% is reached using the second order polynomial pooling. The use of the Gaussian kernel allows to reach 92.74%, whereas the best state-of-the-art method only reports 87% of accuracy [1]. Similarly, with the ExpW dataset and using the proposed method, we outperform state-of-the-art methods with all kernels. We obtained a 76.81% of accuracy using the third order polynomial pooling which exceeds the other methods by more than 3%. Finally, even though we did not outperform state-of-the-art methods on FER2013, we confirmed the superiority of our method compared to standard pooling methods. We reached an accuracy rate of 71.35% with the third order polynomial pooling which is 2% less than state-of-the-art method [15].

4.4. Cross-dataset evaluation

We have also evaluated the generalizability of our network on data from different distributions. We conducted an experiment on a cross-dataset. We compared the performance of our network using the proposed learnable pooling weights with the same network using standard pooling layers. The considered intra-dataset protocol is a training over the whole ExpW dataset and a testing on RAF-DB dataset. The obtained results also confirm the efficiency of the proposed pooling layer in the FER task. Our method using a linear kernel gives 80.27% as accuracy rate. The use of the second-order polynomial kernel allows to achieve 80.56%. The third-order polynomial kernel and the Gaussian RBF kernel give respectively 81.43% and 81.03%. On the contrary, the use of max-pooling layers instead of our learnable weights only allows to reach 80.12%.

5. Conclusion

In this paper, we proposed a FER method based on a CNN model to which we specifically designed a novel pooling layer which retains the down-sampling advantage of an ordinary pooling function and brings several new features. The proposed pooling layer, which has learnable weights, generalizes standard pooling functions and, additionally encodes non-linear relation between features. It is differentiable and can be plugged at any level of the network, allowing, in turns, an end-to-end learning. The experiments on ExpW, RAF-DB and FER2013 datasets demonstrate the efficiency of the proposed pooling method compared to standard pooling. The experiments also showed that the proposed FER method outperforms state-of-the-art methods. The performance of our model is essentially due to its capability of capturing high order information that are crucial for fine-grained classification tasks such as the FER.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. Acharya, Z. Huang, D. Pani Paudel, L. Van Gool, Covariance pooling for facial expression recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 367–374.

[2] D. Acharya, Z. Huang, D. Pani Paudel, L. Van Gool, Covariance pooling for facial expression recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018.

[3] M. Bishay, P. Palasek, S. Priebe, I. Patras, SchiNet: automatic estimation of symptoms of schizophrenia from facial behaviour analysis, IEEE Trans. Affect. Comput. (2019).

[4] J. Cai, Z. Meng, A.S. Khan, Z. Li, J. O'Reilly, Y. Tong, Island loss for learning discriminative features in facial expression recognition, in: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), IEEE, 2018, pp. 302–309.

[5] T. Cohen, M. Welling, Group equivariant convolutional networks, in: International Conference on Machine Learning, 2016, pp. 2990–2999.

[6] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, S. Belongie, Kernel pooling for convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2921–2930.

[7] W. Deng, J. Hu, S. Zhang, J. Guo, DeepEmo: real-world facial expression analysis via deep learning, in: 2015 Visual Communications and Image Processing (VCIP), IEEE, 2015, pp. 1–4.

[8] C. Fabian Benitez-Quiroz, R. Srinivasan, A.M. Martinez, EmotioNet: an accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5562–5570.

[9] Y. Gao, O. Beijbom, N. Zhang, T. Darrell, Compact bilinear pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 317–326.

[10] Z. Gao, L. Wang, G. Wu, Lip: local importance-based pooling, arXiv:1908.04156 (2019a).

[11] Z. Gao, J. Xie, Q. Wang, P. Li, Global second-order pooling convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3024–3033.

[12] I.J. Goodfellow, D. Erhan, P.L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, et al., Challenges in representation learning: a report on three machine learning contests, in: International Conference on Neural Information Processing, Springer, 2013, pp. 117–124.

[13] Y. Guo, D. Tao, J. Yu, H. Xiong, Y. Li, D. Tao, Deep neural networks with relativity learning for facial expression recognition, in: 2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), IEEE, 2016, pp. 1–6.

[14] J. Hyun, H. Seong, E. Kim, Universal pooling–a new pooling method for convolutional neural networks, arXiv: 1907.11440 (2019).

[15] B.-K. Kim, S.-Y. Dong, J. Roh, G. Kim, S.-Y. Lee, Fusing aligned and non-aligned face information for automatic affect recognition in the wild: a deep learning approach, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 48–57.

[16] C.-M. Kuo, S.-H. Lai, M. Sarkis, A compact deep learning model for robust facial expression recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 2121–2129.

[17] S. Li, W. Deng, Deep facial expression recognition: a survey, arXiv: 1804.08348 (2018a).

[18] S. Li, W. Deng, Reliable crowdsourcing and deep locality-preserving learning for unconstrained facial expression recognition, IEEE Trans. Image Process. 28 (1) (2018) 356–370.

[19] S. Li, W. Deng, J. Du, Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, 2017, pp. 2584–2593.

[20] Z. Lian, Y. Li, J.-H. Tao, J. Huang, M.-Y. Niu, Expression analysis based on face regions in read-world conditions, Int. J. Autom. Comput. (2020) 1–12.

[21] T.-Y. Lin, S. Maji, Improved bilinear pooling with CNNs, arXiv: 1707.06772 (2017).

[22] T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNN models for fine-grained visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1449–1457.

[23] X. Liu, B. Vijaya Kumar, J. You, P. Jia, Adaptive deep metric learning for identity-aware facial expression recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 20–29.

[24] Z. Liu, S. Li, W. Deng, Boosting-POOF: boosting part based one vs one feature for facial expression recognition in the wild, in: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), IEEE, 2017, pp. 967–972.

[25] F. Saeedan, N. Weber, M. Goesele, S. Roth, Detail-preserving pooling in deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9108–9116.

[26] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv: 1409.1556 (2014).

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[28] Y. Tang, Deep learning using linear support vector machines, arXiv: 1306.0239 (2013).

[29] J.B. Tenenbaum, W.T. Freeman, Separating style and content with bilinear models, Neural Comput. 12 (6) (2000) 1247–1283.

[30] C. Wang, J. Yang, L. Xie, J. Yuan, Kervolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 31–40.

[31] Y. Wen, K. Zhang, Z. Li, Y. Qiao, A discriminative feature learning approach for deep face recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 499–515.

[32] H. Yang, U. Ciftci, L. Yin, Facial expression recognition by de-expression residue learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2168–2177.

[33] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, A.M. Dobaie, Facial expression recognition via learning deep sparse autoencoders, Neurocomputing 273 (2018) 643–649.
[34] Z. Zhang, P. Luo, C.C. Loy, X. Tang, From facial expression recognition to interpersonal relation prediction, Int. J. Comput. Vis. 126 (5) (2018) 550–569.
[35] Z. Zhang, P. Luo, C.C. Loy, X. Tang, From facial expression recognition to interpersonal relation prediction, arXiv: 1609.06426v2 (2016).
[36] X. Zou, Z. Wang, Q. Li, W. Sheng, Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification, Neurocomputing (2019).