IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2021, accepted August 2, 2021, date of publication August 30, 2021, date of current version September 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108630

Zernike Pooling: Generalizing Average Pooling Using Zernike Moments

THOMAS THEODORIDIS, KOSTAS LOUMPONIAS, NICHOLAS VRETOS, (Member, IEEE), AND PETROS DARAS, (Senior Member, IEEE)

Information Technologies Institute (ITI), Centre for Research and Technology Hellas (CERTH), 57001 Thessaloniki, Greece

Corresponding author: Thomas Theodoridis (tomastheod@iti.gr)

This work was supported by the European Commission (TeNDER, affecTive basEd iNtegrateD carE for betteR Quality of Life) under Grant 875325.

ABSTRACT Most of the established neural network architectures in computer vision are essentially composed of the same building blocks (e.g., convolutional, normalization, regularization, pooling layers, etc.), with their main difference being the connectivity of these components within the architecture and not the components themselves. In this paper we propose a generalization of the traditional average pooling operator. Based on the requirements of *efficiency* (to provide information without repetition), *equivalence* (to be able to produce the same output as average pooling) and *extendability* (to provide a natural way of obtaining novel information), we arrive at a formulation that generalizes average pooling using the Zernike moments. Experimental results on *Cifar 10*, *Cifar 100* and *Rotated MNIST* data-sets showed that the proposed method was able to outperform the two baseline approaches, global average pooling and average pooling 2×2, as well as the two variants of Stochastic pooling and AlphaMEX in every case. A worst-case performance analysis on *Cifar-100* showed that significant gains in classification accuracy can be realised with only a modest 10% increase in training time.

INDEX TERMS Neural networks, pooling, Zernike moments, image classification.

I. INTRODUCTION

Since the introduction of the AlexNet architecture [1] and the wide-ranging impact that it had on computer vision, research in this field has been progressing at an ever-increasing pace, with new architectures being proposed every year. The ZFNet architecture proposed in [2] was an improved version of AlexNet, guided by feature visualization through the use of a deconvolutional network. The Inception architecture (GoogLeNet) [3] was built using stacks of inception modules. The core idea behind the inception module is the parallel processing of its input through four distinct paths that are finally combined. The first path consists of a 1×1 convolutional layer, the second consists of a 1×1 convolutional layer followed by a 3×3 one, the third by a 1×1 followed by a 5×5, while the last one consists of a 3×3 max pooling layer followed by a 1×1 convolutional layer. The VGG network [4] had a straightforward architecture, with a series of convolutional layers with small 3×3 receptive fields followed by max pooling layers and three fully-connected layers at the end. Residual networks (ResNets) [5] introduced shortcut connections every few layers in the architecture and showed that it is possible to train a considerably deeper network with this architecture compared to previous works, reaching up to 1202 layers. Wide residual networks (WRNs) [6] increased the number of features in convolutional layers and decreased the overall depth of the network compared to ResNets. DenseNets [7] improved on the idea of shortcut connections by using the output of a network layer as input to all subsequent layers. This dense connectivity pattern among the network layers allows gradients to propagate more easily and encourages feature reuse. Neural architecture search [8] comprised an interesting approach towards neural network architecture design, with a controller RNN being trained with reinforcement learning in order to generate accurate model architectures for specific tasks. The drawback, however, with this approach was the extremely high computational cost. As an evolution of the last approach, EfficientNets [9] were obtained through a neural architecture search for a baseline model, which can then be efficiently scaled up

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili.

using a specific ratio among its width, depth and resolution attributes.

This progress over the last decade resulted in considerable improvements in model inference accuracy and architecture efficiency, both in terms of parameters and computational cost. The novelty of the different network designs, however, was mostly the result of individual choices regarding the depth of the architecture, the connectivity pattern of the network components as well as the parameter values of these components, such as the receptive field of convolutional layers. The core network components themselves, such as the convolutional and pooling layers, have remained essentially the same across all of the aforementioned architectures. Although there have been efforts towards improving convolutional [10] and pooling [11] layers, this research domain has received far less attention from the research community.

This work focuses on improving one of the core network components, the pooling layer. Given that pooling layers a) are ubiquitous in network architectures used in computer vision and b) act as a mechanism of summarizing the information within the region of pooling, being able to improve them so they could retain more information from each pooling region would be beneficial for the overall performance of the architecture. To this end, we propose in this paper a generalization of the average pooling operator using the Zernike moments [12]. We selected these moments based on the following requirements:

1) **Efficiency**. Zernike polynomials, and by extension Zernike moments, are by construction orthogonal to each other. This means that each additional higher-order moment contributes information that cannot be obtained by previous ones. This property makes the proposed method efficient.
2) **Equivalence**. The first Zernike moment ($A_{0,0}$) computes the average value of the region it is applied to. Therefore, the proposed method provides a way of adjusting its behaviour so that it is equivalent to traditional average pooling.
3) **Extendability**. Higher-order Zernike moments (e.g., $A_{1,1}, A_{2,0}, A_{2,2}$, etc.) constitute a natural way of extending average pooling and providing novel information.

Furthermore, an additional benefit of these moments is that they retain the property of being rotation invariant, like average pooling, across all higher-order moments when their magnitudes are used.

Following is a brief explanation of how the proposed method works. Assuming that Zernike pooling is used in place of global average pooling, each Zernike polynomial needed for the chosen order is multiplied element-wise (Hadamard product) with the input feature map. Then, the resulting representation is averaged individually for each channel, eliminating the spatial dimension of the input, and its magnitude is computed. This produces a feature vector with the same number of channels as the input feature map. Finally, the feature vectors produced by all needed polynomials are concatenated in order to produce the output of our method. The proposed method outperforms traditional global average pooling by more than 1.9% and 1.3% in *Cifar-100* when using different DenseNet variants.

The rest of this paper is organized as follows: Section II provides an overview of related works regarding extensions of the pooling operation and the use of Zernike moments in computer vision. In Section III the mathematical formulation of the proposed method is presented. The experimental evaluation of our method is presented in Section IV. In Section V the limitations of our method are presented. Finally, in Section VI conclusions are drawn.

II. RELATED WORK

A. ZERNIKE MOMENTS

Zernike moments [12] are well-known in the field of pattern recognition due to their desirable properties, such as the rotation invariance of their magnitudes. Most commonly, they have been applied directly to input images, in order to obtain suitable representation for reconstruction and recognition purposes.

Khotanzad and Hong [13] utilized the Zernike moments as image representations which are invariant to rotation, scale and translation. Paired with a nearest-neighbor (NN) classifier, the Zernike representations achieved an almost perfect classification score when applied to a binary image data-set consisting of scaled, translated and rotated upper case English characters, as well as to binary images depicting aerial views of four different lakes under various orientations.

Kan and Srinath [14] compared both Zernike moments and orthogonal Fourier-Mellin moments as image descriptors for the task of English character recognition from binary images. Experimental results under two different pre-processing schemes showed that Zernike moments performed better when samples were normalized for size, slant and stroke, while orthogonal Fourier-Mellin moments performed better in the stroke-only normalization scheme.

Wu *et al.* [15] compared the performance of several types of moments, such as Geometric, Zernike, Legendre, etc., on classifying binary images into 9 classes. In their framework, each moment was applied twice, one directly on the input images and then on the extracted features. The final recognition step was performed by a SVM classifier. The best performance out of all tested moments was achieved with the Zernike ones.

Haddadnia *et al.* [16] proposed a human face recognition method based on image representations obtained from pseudo-Zernike polynomials. First, their method performs human face localization so that only relevant parts of the images are taken into account during feature extraction. Next, pseudo-Zernike moment invariant features from these parts are obtained. Finally, these features are processed by a radial basis function (RBF) neural network classifier. Experimental results on the ORL face database indicated a recognition rate above 99%.

Vretos *et al.* [17] employed Zernike moments for 3D facial expression recognition from depth images. To this end, the initial 3D face point clouds were first converted into depth images. A feature representation consisting of order 20 Zernike moments was then extracted from these images and was used in order to train a support vector machine (SVM) classifier to recognize facial expressions. The reported recognition results were 73% and 61% for the BU3DFE and Bosphorus data-sets.

Tahmasbi *et al.* [18] proposed a computer-aided diagnosis system for classifying benign and malignant breast masses based on Zernike moments. Suspect regions of interest are extracted from mammograms and processed in order to obtain a binary image containing the mass shape, as well as a gray-scale image which contains the mass margin. Zernike moments were used for extracting relevant features from these images, which were subsequently processed by a multi-layer perceptron classifier.

A method for the early diagnosis of Alzheimer's disease was proposed by Gorji and Haddadnia [19] based on pseudo-Zernike moments. In order to distinguish people with mild cognitive impairment from Alzheimer's disease and healthy control groups, pseudo-Zernike moments of order 30 were used for extracting relevant information from magnetic resonance images. These features where then processed by a feed-forward fully-connected classifier.

Zhang *et al.* [20] employed pseudo-Zernike moments for the task of pathological brain detection. Moments of order 19 were used for extracting relevant information from brain images, processed by a support vector machine classifier with an RBF kernel. Experimental results on 3 data-sets showed sensitivity and specificity scores above 98%.

B. POOLING

Zeiler and Fergus [11] replaced the conventional deterministic pooling operation with a stochastic procedure, whereby the activation within each pooling region is randomly chosen according to the probability distribution formed by normalizing the activations within this region. At test time, the authors noted that this procedure introduces noise and degrades performance, so the expected activation was used instead, i.e., the weighted average of the activations in the pooling region, with the weights being the aforementioned probabilities. It is worth noting that this approach is parameter-free.

Yu *et al.* [21] proposed a mixed pooling scheme which combines max and average pooling operations in a stochastic manner. During training, a binary random variable determines which of the two operators (max or average) is active for each feature map during each forward pass and a counter measures how many times each of the two was activated. Since similar issues to [11] were observed during evaluation, the layer becomes deterministic during testing, by choosing the operator with the highest count during training. The authors note that this mixed pooling scheme is beneficial for reducing over-fitting. This method also introduces no additional parameters.

Lee *et al.* [22] proposed two strategies for generalizing traditional pooling operations. The first one is similar to [21], with the difference that the mixing ratio between average and max pooling is now a learnable parameter between 0 and 1. The second strategy makes use of the first one, but instead of mixing predefined pooling functions, the functions themselves are also learned.

An interesting rank-based pooling approach was proposed by Shi *et al.* [23] in the form of three variants, depending on the pooling method: average, weighted and stochastic. In the case of rank-based average pooling, the method computes the average of the top t highest activations. When t equals the pooling region size, it corresponds to average pooling, while a value of $t = 1$ results in max pooling. In the cases of rank-based weighted and stochastic pooling, the rank of each element determines the probability p_i assigned to it.

Simon *et al.* [24] proposed α-pooling, a generalization of average and bilinear pooling methods. The pooling strategy is controlled by parameter α, which is jointly optimized with the rest of the network architecture during training. A value of $\alpha = 1$ corresponds to average pooling, while $\alpha = 2$ is equivalent to bilinear pooling.

Stochastic spatial sampling (S3Pool), proposed by Zhai *et al.* [25], is a variation of the deterministic max pooling operation, where the activations inside a pooling region are still based on the max operator, but the downsampling process is stochastic. During inference the process is substituted by average pooling.

Saeedan *et al.* [26] proposed detail-preserving pooling (DPP), a pooling method with learnable parameters that rewards differences in the input activation values. Two variants were tested in terms of reward, one that preserves all details and one that preserves only details with a higher-than-average activation. Based on the ResNet-101 and VGG-16 architectures, the increase in parameters that this method introduced was 43k and 18k respectively, while the computational cost was +5% and +20%.

Zhang *et al.* [27] proposed a smooth, non-linear pooling function (AlphaMEX) which adjusts its behaviour based on the value of a controlling parameter α. For $\alpha = 0$ the function outputs the minimum value within the pooling region, for $\alpha = 1/2$ the average value and for $\alpha = 1$ the maximum value. Parameter α was added to the learnable parameters of the network architecture, biased to lie within $\left(\frac{1}{2}, 1\right)$.

As an alternative to traditional spatial pooling methods which operate on $N \times N$ square regions, Hou *et al.* [28] proposed strip pooling which operates on long but narrow regions $N \times 1$ and $1 \times N$. The output of the horizontal and vertical pooling modules is combined in order to obtain the final representation.

Spatial pyramid pooling, introduced by Grauman and Darrell [29] and Lazebnik *et al.* [30], and popularized in deep neural networks for computer vision by He *et al.* [31], partitions feature map responses into a number of spatial bins of different sizes. Bin sizes are proportional to the input

feature map size and their number is fixed, producing in this way feature vectors of standard dimensions regardless of the input spatial dimension. Chen *et al.* [32], [33] combined this idea with atrous convolutions of different rates in order to capture information at multiple scales. Another variation of the pyramid pooling idea was proposed by Huang *et al.* [34], where the output representations are not converted into a feature vector but are instead concatenated while retaining their spatial dimension in order to produce the final representation.

Expansion downsampling scaling (EDS) pooling was proposed by Singh *et al.* [35] in order to minimize the information discarded using traditional pooling approaches such as max and average pooling. EDS first increases the number of input channels, while simultaneously reducing the input spatial dimension (expansion and downsampling). Then it utilizes a learnable scaling process in order to adjust the importance of the extracted features.

Williams and Li [36] proposed wavelet pooling, a novel pooling approach that uses wavelets to reduce the spatial dimension of feature maps, in contrast to traditional neighbour-based approaches. The method performs a second order decomposition in the wavelet domain and constructs the reduced representation using only the second order wavelet subbands, while discarding the first order fine-scale coefficients. Wolter and Garcke [37] improved upon the idea of [36] by introducing adaptive and scaled wavelet pooling, which outperformed the former method in all experiments.

Finally, the survey of Akhtar and Ragavendran [38] on this topic provides a breakdown of the various pooling methods into general categories, as well as detailed explanations and illustrations.

III. METHOD DESCRIPTION

The general description and mathematical formulation of Zernike moments is presented in subsection III-A, while in subsection III-B the proposed method for generalizing average pooling is presented.

A. ZERNIKE MOMENTS

In the general continuous case, the complex Zernike moments of order n with repetition m of a function $f(r,\theta)$ are defined on the unit disc as follows:

$$A_{n,m} = \frac{n+1}{\pi}\int_0^{2\pi}\int_0^1 f(r,\theta)Z^*_{n,m}(r,\theta)r\ drd\theta, \tag{1}$$

where (r,θ) are defined over the unit disc, $n \in \mathbb{N}$ and $m \in \mathbb{Z}$, such that the following conditions hold:

$$n - |m| \text{ is even}, \ \ |m| \leq n \tag{2}$$

$Z_{n,m}(r,\theta)$ represents the Zernike polynomial of order n with repetition m and is defined (in polar coordinates) as:

$$Z_{n,m}(r,\theta) = R_{n,m}(r)exp(i\cdot m\cdot\theta), \tag{3}$$

where $i = \sqrt{-1}$ and $R_{n,m}(r)$ is the orthogonal radial polynomial defined as:

$$R_{n,m}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} \frac{(-1)^s(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}r^{n-2s} \tag{4}$$

$R_{n,m}(r) = R_{n,-m}(r)$ and it must be noted that if the conditions in (2) are not met, then $R_{n,m}(r)$ is equal to 0.

In the discrete case, let I be a matrix with spatial dimension $M \times N$ (without loss of generality, in the rest of this work we will refer to I as matrix, image or feature map interchangeably, despite the fact that the last two are $M \times N \times D$ tensors, since the process is applied to each channel D independently). $I(k,j)$ represents the value of the matrix located at position (k,j). Then, the Zernike moments are defined as:

$$A_{n,m} = \frac{n+1}{\pi}\sum_0^{M-1}\sum_0^{N-1} I(k,j)Z^*_{n,m}(r,\theta) \tag{5}$$

where $Z^*_{n,m}(r,\theta)$ is the complex conjugate of $Z_{n,m}(r,\theta)$ and the discrete polar coordinates are given as:

$$\begin{aligned} r_{k,j} &= \sqrt{x_j^2 + y_k^2}, \\ \theta_{k,j} &= arctan(\frac{y_k}{x_j}) \end{aligned} \tag{6}$$

while x_j and y_k are computed in the following way:

$$\begin{aligned} x_j &= c + \frac{j\cdot(d-c)}{N-1} \\ y_k &= c - \frac{k\cdot(d-c)}{M-1} \end{aligned} \tag{7}$$

The numbers c and d are equal to -1 and 1, respectively, in the case of mapping a matrix outside of the unit circle (parts of the matrix are not mapped), while they are equal to $-1/\sqrt{2}$ and $1/\sqrt{2}$, respectively, in the case of mapping the matrix inside the circle (the circle circumscribes matrix I).

Zernike moments can also be used in order to reconstruct matrix I:

$$\widehat{I}(r,\theta) = \sum_{n=0}^{n_{max}}\sum_m A_{n,m}Z_{n,m}(r,\theta) \tag{8}$$

where $\widehat{I}(r,\theta)$ is the reconstructed matrix and n_{max} is the maximum order of the moments used for the reconstruction process. The reconstruction $\widehat{I}$ can come arbitrarily close to the original I as $n_{max} \to \infty$.

B. ZERNIKE POOLING

Assuming tensor $I \in \mathbb{R}^{M\times N\times D}$ represents a real-valued image or feature map with M rows, N columns and D channels, by utilizing an appropriate radius for its projection onto the Zernike basis functions (such that the unit circle circumscribes it, as mentioned in III-A), the Zernike moments of order n are calculated using Equation (5). The resulting representation is:

$$I'_{[n]} = \left[|A_{n,m_1}|\ldots|A_{n,m_\omega}|\right] \in \mathbb{R}^{D\times l_n} \tag{9}$$

where $|\dots|$ denotes the magnitude, $[\dots]$ denotes concatenation and l_n denotes the number of order n moments that exist. In other words, $I'_{[n]}$ represents the concatenation of the magnitudes of all l_n moments of order n into a single matrix of size $D \times l_n$. Consequently, in order to represent tensor I with Zernike moments up to order n, the magnitudes of all moments up to n are concatenated:

$$I'_n = \left[I'_{[0]} I'_{[1]} \dots I'_{[n]}\right] = \left[|A_{0,0}||A_{1,1}| \dots |A_{n,m_1}| \dots |A_{n,m_\omega}|\right] \in \mathbb{R}^{D \times L_n} \quad (10)$$

where $L_n = l_0 + \dots + l_n$. It is evident that the size of representation I'_n increases by l_{n+1} for each additional order $n+1$.

Finally it is worth noting that the aforementioned representation I'_n can be used to describe the whole feature map I or just smaller spatial regions therein. The first case would be similar to the global average pooling operation, while the second one would correspond to per-region average pooling.

IV. EXPERIMENTAL EVALUATION

A. EXPERIMENTAL SET-UP

The proposed pooling method was evaluated on the *Cifar-10*, *Cifar-100* [39] and *Rotated MNIST* data-sets [40]. Cifar consists of 60000 colour images of size 32×32, split into 50000 images for training and 10000 for testing. As indicated by the name, images in *Cifar-10* belong to 10 classes, while in *Cifar-100* to 100. *Rotated MNIST* is a heavily modified version of MNIST, where each digit was randomly rotated by $\theta \in (0, 2\pi)$ radians. Since this data-set is split into 12000 images of size 28×28 for training and 50000 for testing, it provides an opportunity to evaluate the generalization performance of our method under limited training data.

Two variants of the DenseNet [7] architecture were employed in our experiments, DenseNet-40 (1M params) and DenseNet-22 (300K params), in order to assess the effectiveness of Zernike pooling on architectures of different sizes. The global average pooling (GAP) layer within each network, placed before the fully-connected classifier, was replaced by global Zernike pooling (GZP), thus utilizing the proposed pooling layer only once in each architecture. Since the concatenation formulation (Eq. 10) was used for the Zernike moments, an input feature map with D channels would increase in size to $D \times L_n$ after Zernike pooling. Therefore, a second baseline approach was created by replacing global average pooling with 4×4 average pooling, in order to produce a feature map of size $2 \times 2 \times D$, which was finally reshaped to $4 \times D$. This approach was named average pooling 2×2 (AP2) and corresponds, in terms of overall network parameters, with order 2 Zernike pooling (GZP_2). In addition to these baselines, we also implemented two variants of Stochastic pooling [11] for comparison purposes. The first one (SP) follows the approach of [11], while the second is similar to our baselines, i.e. global Stochastic pooling (GSP) replaces global average pooling in the architecture. Since the activations reaching the pooling layers in DenseNets cannot be assumed to be positive, as was the case in [11], they are converted into probabilities using the *softmax* function. Finally, results from [27] (AlphaMEX pooling) are also reported in the DenseNet-40 case for comparison.

All networks were trained using stochastic gradient descend (SGD) with an initial learning rate of 0.1 and a momentum of 0.9. Learning rate was divided by 10 after 50% and 75% of the total training epochs had elapsed. In the case of the smaller DenseNet-22 architecture the total number of training epochs was scaled by 0.66, resulting in 200 epochs on *Cifar* and 66 on *Rotated MNIST*, while DenseNet-40 was trained for 300 and 100 respectively. Batch size was set to 64 and no data augmentation process was used.

B. EXPERIMENTAL RESULTS

1) EVALUATION OF DIFFERENT ORDERS FOR ZERNIKE POOLING

In order to evaluate the impact of using different orders for Zernike pooling in terms of accuracy and training time, we provide experimental results on *Cifar 100* with both architectures in Table 1, using moments up to order 4. It is worth noting that the choice of *Cifar 100* for this evaluation, the most demanding data-set of the three, constitutes a worst-case scenario for our method in terms of parameters and time, since the concatenation formulation for Zernike moments (Eq. 10) is combined with a high number of output classes.

TABLE 1. Classification results on *Cifar 100* with the DenseNet-22 and DenseNet-40 architectures using higher-order moments for Zernike pooling.

	DenseNet-22			DenseNet-40		
Method	Time	Params	Acc	Time	Params	Acc
GAP	1	297K	67.07	1	1.08M	72.34
GZP_1	1.05	320K	68.43	1.04	1.12M	73.03
GZP_2	1.10	366K	68.24	1.07	1.21M	**73.66**
GZP_3	1.12	413K	68.35	1.09	1.30M	73.35
GZP_4	1.18	482K	**69.00**	1.13	1.44M	73.65

In the case of DenseNet-22, we see that GZP_1 provided a significant increase in performance compared to GAP with only a 5% increase in training time, while a further improvement was observed with GZP_4, reaching an accuracy of 69% with an increase of 18% in training time. We also tested 5th order moments, but they resulted in significant over-fitting to the training data. In the case of the DenseNet-40 architecture, GZP_1 provided a 0.69 improvement in accuracy for a 4% increase in training time compared to GAP. 2nd order moments doubled the performance improvement offered by the previous ones, reaching a 73.66% accuracy with an increase of 7% in training time. Higher-order moments provided similar performance with up to a 13% increase in training time. Based on these findings,

Zernike moments up to order 2 were chosen for the comparison with other pooling methods, as they offer a good trade-off between accuracy gain and increase in training time.

The train and test loss curves of the GAP, GZP_2 and GZP_4 methods with the DenseNet-40 architecture on *Cifar 100* can be seen in Figure 1. As we can observe, the train loss curves of GZP_2 and GZP_4 immediately drop below the GAP one at the beginning of training and remain at this configuration ($GZP_4 < GZP_2 <$ GAP) until the end of training. The test loss curves follow a similar pattern, but their behaviour can be observed more clearly after the first learning rate decrease at epoch 150. These results, in combination with the classification results of Table 1, indicate that the proposed method is able to extract novel information from the input features compared to GAP, which in turn is translated into higher classification accuracy. When comparing GZP_4 to GZP_2, we observe that the additional information offered by the two extra moments could not be translated into higher classification accuracy in this instance. Instead, it was translated into larger classification margins.

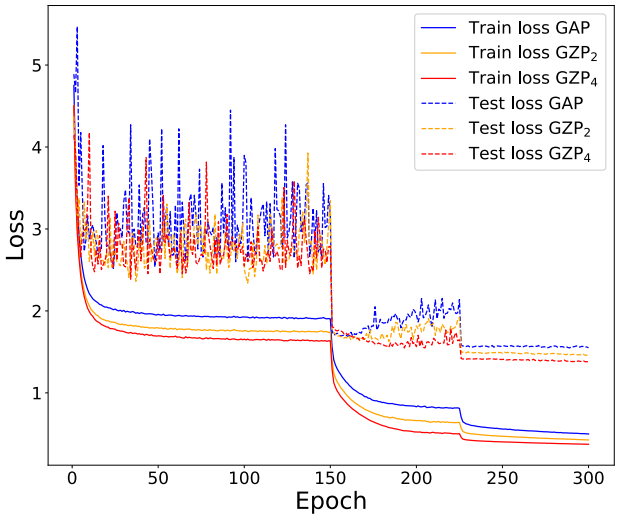


FIGURE 1. DenseNet-40 train and test losses on *Cifar 100* using GAP (blue), GZP_2 (orange) and GZP_4 (red).

2) EVALUATION OF POOLING METHODS

Experimental results with the DenseNet-40 architecture are presented in Table 2. GAP [7] denotes results reported by Huang *et al.*, while GAP denotes our evaluation of the same method. Starting off with *Cifar-10*, the proposed Zernike pooling with up to order 1 moments (GZP_1) achieved an accuracy of 93.65, finishing ahead of GAP by 0.31, surpassing GSP and SP by 0.79 and 1.4 respectively, as well as AlphaMEX by 0.19. Results in the more challenging *Cifar-100* data-set follow a similar pattern, but differences among the methods grew wider here. GZP_2 managed to outperform GAP by 1.32 and AlphaMEX by 0.9, achieving an overall score of 73.66. The difference to SP and GSP grew substantially here, to more than 4 and 6 accuracy points respectively. In *Rotated MNIST* GZP_2 provided an improvement of 1.53, 1.86, 3.73 and 4.35 compared to GAP, SP, GSP and AP2 respectively, reaching an accuracy of 97.85.

TABLE 2. Classification results on *Cifar 10*, *Cifar 100* and *Rotated MNIST* with the DenseNet-40 architecture.

Method	*Cifar 10* Acc	*Cifar 100* Acc	*Rotated MNIST* Acc
GAP [7]	93.00	72.45	–
GAP	93.34	72.34	96.32
AP2	92.69	68.57	93.50
SP	92.25	69.62	95.99
GSP	92.86	67.31	94.12
AlphaMEX [27]	93.46	72.76	–
GZP_1	**93.65**	**73.03**	**97.01**
GZP_2	93.07	**73.66**	**97.85**

Regarding the effectiveness of Zernike pooling on smaller architectures, Table 3 summarizes the results for the DenseNet-22 case. GZP_1 is ahead of GAP by 0.14, AP2 by 0.64 and surpasses SP and GSP by more than 1 point in *Cifar-10*. The performance advantage of our method in *Cifar-100* increases substantially, reaching 1.36, 1.84, 4.57 and 12.64 higher than GAP, AP2, SP and GSP. In the case of *Rotated MNIST*, GZP_1 was 0.58 ahead of AP2 and 1.53 ahead of GAP. Compared to SP and GSP it achieved an improvement of 2.87 and 5.53 respectively. GZP_2 improved upon the first order results by 0.19 in *Cifar-10* and by 0.07 in *Rotated MNIST*, while it performed slightly worse in *Cifar-100* (-0.19). These results are in accordance with the DenseNet-40 case, indicating that Zernike pooling is effective in both deep and shallow architectures.

TABLE 3. Classification results on *Cifar 10*, *Cifar 100* and *Rotated MNIST* with the DenseNet-22 architecture.

Method	*Cifar 10* Acc	*Cifar 100* Acc	*Rotated MNIST* Acc
GAP	91.06	67.07	95.88
AP2	90.56	66.59	96.83
SP	89.68	63.86	94.54
GSP	90.05	55.79	91.88
GZP_1	**91.20**	**68.43**	**97.41**
GZP_2	**91.39**	**68.24**	**97.48**

3) COMPUTATIONAL COMPLEXITY

In this section we analyze the time complexity of Zernike and average pooling in order to complement the experimental results presented in Table 1, as well as to provide a concrete understanding of the computational efficiency of each algorithm (in terms of the number of performed operations) with respect to the input size. To this end, we make use of the big-O notation [41]. Given functions $f(n)$ and $g(n)$ from $\mathbb{N} \rightarrow \mathbb{N}$, $f(n) = O(g(n))$ if there exists a constant c such that $f(n) \leq cg(n)$ for $n \geq n_0$. Furthermore, in order to put the computational cost of the pooling modules into perspective, we also compute the num-

ber of floating point operations (FLOPs) performed by the rest of the network components. Other works concerned with the efficiency of deep neural networks [42]–[44] have also employed similar metrics (FLOPs, fused multiply-adds, etc.). In this work, we assume that multiplications and additions are not fused and that each basic operation and evaluation of elementary functions requires one floating point operation.

Without loss of generality, we assume that the input feature map $I \in \mathbb{R}^{N\times N\times D}$ for both pooling modules has N rows and columns and D channels. In the case of Zernike pooling, the L_n Zernike polynomials $Z^*_{n,m} \in \mathbb{C}^{N\times N}$ needed for order n pooling are also given to the network as input. Regarding the time complexity of global average pooling, for each channel D, we have to sum the values of I across both spatial dimensions and then divide the result by the size of the corresponding area. Therefore, for each channel D, we have an inner complexity of $O(N^2)$, resulting in an overall complexity of $O(DN^2)$. Regarding the time complexity of global Zernike pooling of order n, for each of the L_n Zernike polynomials and for each channel D, we first compute the Hadamard product between polynomial $Z^*_{n,m}$ and the input feature map I, then, we compute the global average of the resulting feature map, obtaining Zernike moment $A_{n,m}$, and finally, we compute the magnitude of $A_{n,m}$. The first two operations involve traversing all $N \times N$ elements of the corresponding matrices and performing basic operations, while the last one involves basic operations and calculating the square root of a single number. Therefore, the inner complexity is dominated by $O(N^2)$, resulting in an overall complexity of $O(L_nDN^2)$. In the special case where order n is given (fixed), L_n can be considered as a constant c and we can say that both pooling methods belong to the same complexity class of $O(DN^2)$. Finally, it is worth mentioning that although the steps performed by Zernike pooling involve complex numbers, and therefore require more operations to compute than their real counterparts (e.g., complex multiplication requires 4 multiplications and 2 additions), nevertheless they still belong to the same complexity classes as their real counterparts.

In order to better understand the computational cost of the pooling modules in relation to the total number of floating point operations (FLOPs) performed by the network, Table 4 shows the FLOPs of different network components for two separate architectures (DenseNet-22 and DenseNet-40) employed on *Cifar 100*. For each architecture, three different settings are shown in the table. All three share the same base model, which contains all network layers before global pooling, and differ only in the pooling module used and the final fully-connected (FC) layer, as Zernike pooling affects the number of features reaching it. As we can see, the operations performed by the base model account for more than 99% of the total number of FLOPs performed by the whole network in all cases, reaching as high as 99.98% in the case of DenseNet-40 with GAP. Although not shown in the table, the operations within the base model are dominated by convolutional layers, which account for more than 96% of the total FLOPs of the network. Regarding the pooling and fully-connected layers, the combination GAP + FC contributes $0.02 - 0.04\%$ to the total number of FLOPs, GZP_2 + FC contributes $0.24-0.44\%$ depending on the architecture, while GZP_4 + FC contributes 0.53% in the case of DenseNet-40 and almost 1% in the case of DenseNet-22. When comparing the difference in FLOPs between DenseNet-22 with GAP and with GZP_4, an almost 1% increase in total FLOPs, and the recorded 18% increase in training time for the same pair in Table 1, we see that the added operations by Zernike pooling have a disproportionate effect in training time. This indicates potential inefficiencies in our implementation of Zernike pooling that can be improved. For instance, the calculation of each Zernike order could be improved so that it runs in parallel, instead of running sequentially after the previous order. This would not affect the number of FLOPs, but would result in lower overhead in training time.

TABLE 4. Number of floating point operations (FLOPs) for different components of the DenseNet-22 and DenseNet-40 architectures on *Cifar 100*.

	DenseNet-22		DenseNet-40	
Component	FLOPs	% of total	FLOPs	% of total
Base Model	149.5M	99.96	541.5M	99.98
GAP	15K	0.01	29K	< 0.01
FC	47K	0.03	90K	0.02
Base Model	149.5M	99.56	541.5M	99.76
GZP_2	479K	0.32	925K	0.17
FC	186K	0.12	359K	0.07
Base Model	149.5M	99.01	541.5M	99.47
GZP_4	1M	0.71	2M	0.38
FC	418K	0.28	807K	0.15

V. LIMITATIONS

Regarding the limitations of the proposed method, although the experimental results in Section IV showed that Zernike pooling provides better performance than competing methods, the use of higher order moments does not always translate to increased performance. Furthermore, there exists a limit in the number of orders that can be effectively utilized with the concatenation formulation, after which the network starts to significantly over-fit to the training data if counter measures are not employed.

Section IV also discussed the effect of Zernike pooling in training time, showing an increase between 4% and 18% depending on the order and network architecture. When these results are considered together with the ones of Table 4, they suggest an inefficiency in our implementation of Zernike pooling that could be improved in the future.

VI. CONCLUSION

The focus of this paper was on improving one of the core components of state-of-the-art architectures in computer vision, the pooling layer. Based on the requirements of *efficiency*,

equivalence and *extendability*, a novel pooling method was proposed that generalizes average pooling using the Zernike moments. Experimental results on *Cifar-10*, *Cifar-100* and *Rotated MNIST* data-sets with the DenseNet-22 and DenseNet-40 architectures showed that the proposed method was able to outperform the two baseline approaches, GAP and AP2, as well as the two variants of Stochastic pooling and AlphaMEX in every case. A time complexity analysis of Zernike pooling showed that when the order n is given, then both Zernike and average pooling belong to the same complexity class of $O(DN^2)$.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 818–833.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[6] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: http://arxiv.org/abs/1605.07146

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[8] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–16.

[9] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114.

[10] M. Weiler, F. A. Hamprecht, and M. Storath, "Learning steerable filters for rotation equivariant CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 849–858.

[11] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, *arXiv:1301.3557*. [Online]. Available: http://arxiv.org/abs/1301.3557

[12] M. R. Teague, "Image analysis via the general theory of moments," *J. Opt. Soc. Amer.*, vol. 70, no. 8, pp. 920–930, 1980.

[13] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 5, pp. 489–497, May 1990.

[14] C. Kan and M. D. Srinath, "Invariant character recognition with Zernike and orthogonal Fourier–Mellin moments," *Pattern Recognit.*, vol. 35, no. 1, pp. 143–154, Jan. 2002.

[15] J. Wu, S. Qiu, Y. Kong, Y. Chen, L. Senhadji, and H. Shu, "MomentsNet: A simple learning-free method for binary image recognition," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2667–2671.

[16] J. Haddadnia, M. Ahmadi, and K. Faez, "An efficient feature extraction method with pseudo-Zernike moment in RBF neural network-based human face recognition system," *EURASIP J. Adv. Signal Process.*, vol. 2003, no. 9, pp. 890–901, Dec. 2003.

[17] N. Vretos, N. Nikolaidis, and I. Pitas, "3D facial expression recognition using Zernike moments on depth images," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 773–776.

[18] A. Tahmasbi, F. Saki, and S. B. Shokouhi, "Classification of benign and malignant masses based on Zernike moments," *Comput. Biol. Med.*, vol. 41, no. 8, pp. 726–735, Aug. 2011.

[19] H. T. Gorji and J. Haddadnia, "A novel method for early diagnosis of Alzheimer's disease based on pseudo Zernike moment from structural MRI," *Neuroscience*, vol. 305, pp. 361–371, Oct. 2015.

[20] Y.-D. Zhang, Y. Jiang, W. Zhu, S. Lu, and G. Zhao, "Exploring a smart pathological brain detection method on pseudo Zernike moment," *Multimedia Tools Appl.*, vol. 77, pp. 22589–22604, Sep. 2018.

[21] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Proc. Int. Conf. Rough Sets Knowl. Technol.* Cham, Switzerland: Springer, 2014, pp. 364–375.

[22] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Proc. Artif. Intell. Statist.*, 2016, pp. 464–472.

[23] Z. Shi, Y. Ye, and Y. Wu, "Rank-based pooling for deep convolutional neural networks," *Neural Netw.*, vol. 83, pp. 21–31, Nov. 2016.

[24] M. Simon, Y. Gao, T. Darrell, J. Denzler, and E. Rodner, "Generalized orderless pooling performs implicit salient matching," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4960–4969.

[25] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris, "S3Pool: Pooling with stochastic spatial sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4970–4978.

[26] F. Saeedan, N. Weber, M. Goesele, and S. Roth, "Detail-preserving pooling in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9108–9116.

[27] B. Zhang, Q. Zhao, W. Feng, and S. Lyu, "AlphaMEX: A smarter global pooling method for convolutional neural networks," *Neurocomputing*, vol. 321, pp. 36–48, Dec. 2018.

[28] Q. Hou, L. Zhang, M.-M. Cheng, and J. Feng, "Strip pooling: Rethinking spatial pooling for scene parsing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4003–4012.

[29] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, 2005, pp. 1458–1465.

[30] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2169–2178.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[32] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[33] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*. [Online]. Available: http://arxiv.org/abs/1706.05587

[34] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection," *Inf. Sci.*, vol. 522, pp. 241–258, Jun. 2020.

[35] P. Singh, P. Raj, and V. P. Namboodiri, "EDS pooling layer," *Image Vis. Comput.*, vol. 98, Jun. 2020, Art. no. 103923.

[36] T. Williams and R. Li, "Wavelet pooling for convolutional neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[37] M. Wolter and J. Garcke, "Adaptive wavelet pooling for convolutional neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 1936–1944.

[38] N. Akhtar and U. Ragavendran, "Interpretation of intelligence in CNN-pooling processes: A methodological survey," *Neural Comput. Appl.*, vol. 32, no. 3, pp. 879–898, Feb. 2020.

[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[40] *Rotated MNIST Data-Set*. Accessed: Aug. 8, 2021. [Online]. Available: https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits

[41] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

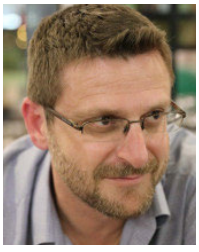
[42] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–13.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. A. Gomez, L. Kaiser, and P. Illia, "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.
[44] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

THOMAS THEODORIDIS received the bachelor's (B.Sc.) degree in mathematics and the master's (M.Sc.) degree in web science from the School of Mathematics, Aristotle University of Thessaloniki, Greece. He has participated in five European projects as a Researcher. His research interests include pattern recognition, computer vision, neural networks, human action recognition, and complex systems.

KOSTAS LOUMPONIAS received the B.Sc. degree in mathematics, and the M.Sc. and Ph.D. degrees in statistics from the School of Mathematics, Aristotle University of Thessaloniki, Greece, in 2013, 2015, and 2020, respectively. He is currently working as a Postdoctoral Researcher with the Centre for Research and Technology Hellas. His main research interests include hidden states estimation with censored data, social network analysis, and neural networks.

NICHOLAS VRETOS (Member, IEEE) received the B.Sc. degree in computer science from the University Pierre et Marie Curie (Paris VI), in 2002, and the Ph.D. degree from the Aristotle University of Thessaloniki, in 2012. He is currently a Researcher at VCL/ITI-CERTH. He worked in twelve European projects as a Technical Manager/WP Leader/Researcher. He has published more than 70 articles in scientific journals and conference proceedings and a book chapter. He has committed as a reviewer for several journals and conferences in the field of image and video processing. His main interests include image and video processing, semantic analysis, neural networks, and 3-D data processing.



PETROS DARAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 1999, 2002, and 2005, respectively. He is currently a Research Director and the Chair of the Visual Computing Laboratory, Information Technologies Institute. He has been involved in more than 50 projects, funded by the EC and the Greek Ministry of Research and Technology. His main research interests include visual content processing, multimedia indexing, search engines, recommendation algorithms, and relevance feedback. His involvement with those research areas has led to the coauthoring of more than 300 articles in refereed journals and international conferences.