A novel technique for optimizing the filter size of CNNs without backpropagation

Muhammad Manzar Maqbool
Electronic Engineering Department
Royal Holloway University of London
Egham, UK
Mohammed.Manzar.2019@live.rhul.ac.uk

Clive Cheong Took
Electronic Engineering Department
Royal Holloway University of London
Egham, UK
Clive.CheongTook@rhul.ac.uk

Saeid Sanei
Computer Science Department
Nottingham Trent University
Nottingham, UK
saeid.sanei@ntu.ac.uk

***Abstract*—Image filters play a crucial role in the performance of convolutional neural networks (CNNs). Yet, the optimisation of those filters tends to focus solely on optimising their weights. As a result, the machine learning practitioner either uses the standard 3×3 filter size or has to select the filter size empirically, as the optimal filter size depends on the application at hand. There has been a lack of serious attempts to address this issue in CNNs. To this end, we propose a novel technique to learn the filter size without depending on either gradient descent or backpropagation. We compare our technique with the only serious attempt existing and show that not only our method performs better, but also converges to an optimal solution at a much smaller number of training iterations.**

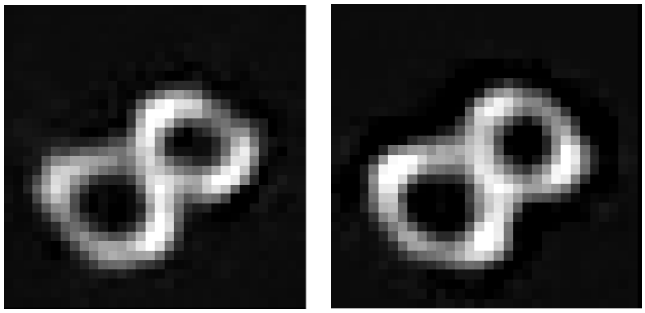
***Index Terms*—Back propagation, CNN, deep learning, filter size, receptive field.**

I. Introduction

Convolutional neural networks (CNNs) have become deeper over time. For example, ResNet [1], which was the champion of ILSVRC 2015, is around 8 times deeper than VGGNet [2] and 20 times deeper than AlexNet [3]. Deeper networks usually tend to improve the performances, yet increasing depth of a network also makes it more complex and difficult to optimize.

Among the challenges in optimizing a deep neural network lies the selection of a suitable filter size for its convolutional layers. In CNNs, the filter size defines the receptive field from which the information is captured in a convolutional layer. In traditional settings, most CNNs have a fixed and predetermined filter size for their convolutional layers. However, such a predefined filter size is not necessarily optimal for a given image recognition task. For example, in facial recognition tasks AU-12 needs a larger receptive field than AU-17 [4], thus, a larger filter size should be more suitable for its detection. For instance, a larger filter size enhances the reconstruction of an image as shown in Fig. 1. However, a larger filter size can also capture more noise, which may lead to a degradation in performance. As such, there is a need to find an optimal filter size.

For a given input size, the optimal filter size for each convolutional layer is often chosen experimentally or via visualization [5]. For example, the recipient of the best expression recognition performance of Emoti 2015 challenge [6] selected the filter size for their convolutional layers experimentally. However, due to the increasing depth of CNNs [1] [7], it is infeasible to search for the filter size exhaustively due to vastly expensive training costs.



(a) (b)

Fig. 1. An increase in filter size from 3×3 in (a) to 9×9 in (b) enhances the image reconstruction.

In order to address these difficulties, we propose a novel method which learns the filter size of a single convolutional layer without relying on back propagation. This work also evaluates Han *et al.*'s method [4] for regression based image denoising, and compares its performance with the proposed method.

The rest of this paper is organised as follows. Section II gives details about our technique and its derivation, whereas Section III discusses the simulation results ending with the conclusion in Section IV.

II. System model

The Variable Tap Length Algorithm (VTLA), which was introduced to find the optimum tap-length of adaptive filters [8] [9], is considered to optimize filter size of CNNs in this work. As such, the proposed method is referred to as the Variable Filter Length Algorithm (VFLA) for CNNs. For fair comparison, Han et al's algorithm [4] is used as a benchmark algorithm and is referred to as the Optimized Filter Size CNN

978-1-6654-5245-8/23/$31.00 ©2023 IEEE

(OFS-CNN [4]).

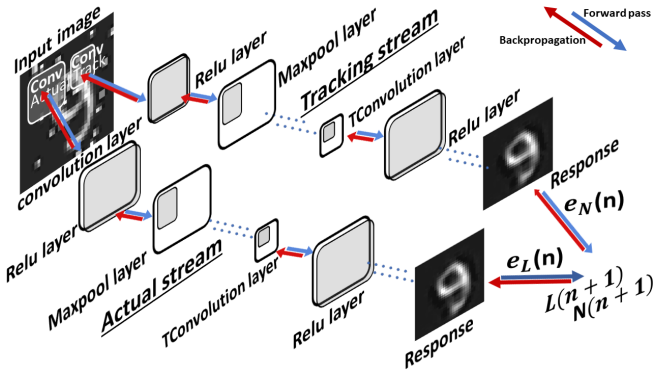


Fig. 2. Visual description of VFLA.

A. Variable Filter Length Algorithm

Unlike VTLA that optimizes one-dimensional filters [8], the proposed VFLA optimizes two-dimensional (2D) filters. Since the VTLA computes the optimal value based on the difference between a pair of errors, VFLA adds a parallel stream of layers to the original CNN architecture, with each layer having a smaller filter size. This is the tracking stream. It lets us compute a pair of regression outputs and mean square errors (MSE) as follows:

$$e_L(n) = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(\mathbf{T}_{i,jL}^{(p)}(n) - \mathbf{X}_{i,jL}^{(p)}(n)\right)^2 \quad (1)$$

$$e_N(n) = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=i}^{I}\sum_{j=1}^{J}\left(\mathbf{T}_{i,jN}^{(p)}(n) - \mathbf{X}_{i,jN}^{(p)}(n)\right)^2 \quad (2)$$

where, for each mini-batch of P images having $I \times J$ pixels and at the nth iteration, $\mathbf{T}$ are the target and $\mathbf{X}$ are the predicted images respectively. Similarly, the subscript $(.)_L$ represents values related to the *actual* filter size L and the subscript $(.)_N$ represents values related to the *tracking* filter size N. It should also be noted that $1 \leq N \leq L$ and $N = L - \Delta$ where Δ is a small positive integer usually set to 2.

These two errors can then be used to calculate the tap length in the form of fractional filter length f as follows:

$$f(n+1) = f(n) - \lambda[e_L(n) - e_N(n)] \quad (3)$$

where λ is the learning rate. The fractional length is hard constrained to remain unchanged if the difference between the errors falls and remains under a threshold ζ:

$$f(n+1) \equiv \begin{cases} f(n) & |e_L(n) - e_N(n)| \leq \zeta \\ f(n+1) & \text{otherwise} \end{cases} \quad (4)$$

The fractional length is not guaranteed to be an integer. This is a problem since filter size of a convolution layer can not be non-integer. In order to address this issue, the filter size L is updated as follows:

$$L(n) \equiv \begin{cases} \text{floor}(f(n)) & |e_L(n) - e_N(n)| > \delta \\ L(n) & \text{otherwise} \end{cases} \quad (5)$$

where floor(.) is used to round down to the nearest integer and δ is a tunable positive integer.

After the computation of filter sizes in each iteration, the filter weights need to be updated as determined by the mini batch Stochastic Gradient Descent (SGD) algorithm with Adaptive Moment Estimation (ADAM) [10]. Because the filter sizes might be greater than or less than their values from the previous iteration, two policies namely VFLA1 and VFLA2 are next proposed to expand or contract the 2D filters accordingly.

For VFLA1 we simply pad zeros at the right and bottom of the matrix to expand it as per L if L increases , and truncate from the same sides to contract it if L decreases. The tracking filter is updated in the same way according to the changes in filter size N.

TABLE I
ARCHITECTURE OF THE BASELINE CNN BASED AUTO-ENCODER

Serial	Name	Type	Activations	Learnables
1	Input	Image Input	$32 \times 32 \times 1$	-
2	Conv1	Convolution	$32 \times 32 \times 16$	weights:$3 \times 3 \times 1 \times 16$ bias:$1 \times 1 \times 16$
3	Relu1	Relu	$32 \times 32 \times 16$	-
4	Maxpool1	Maxpool	$16 \times 16 \times 16$	-
5	Conv2	Convolution	$16 \times 16 \times 8$	weights:$3 \times 3 \times 16 \times 8$ bias:$1 \times 1 \times 8$
6	Relu2	Relu	$16 \times 16 \times 8$	-
7	Maxpool2	Maxpool	$8 \times 8 \times 8$	-
8	Conv3	Convolution	$8 \times 8 \times 8$	weights:$3 \times 3 \times 8 \times 8$ bias:$1 \times 1 \times 8$
9	Relu3	Relu	$8 \times 8 \times 8$	-
10	Maxpool3	Maxpool	$4 \times 4 \times 8$	-
11	Tconv1	Transposed Convolution	$8 \times 8 \times 8$	weights:$4 \times 4 \times 8 \times 8$ bias:$1 \times 1 \times 8$
12	Relu4	Relu	$8 \times 8 \times 8$	-
13	Tconv2	Transposed Convolution	$16 \times 16 \times 8$	weights:$4 \times 4 \times 8 \times 8$ bias:$1 \times 1 \times 8$
14	Relu5	Relu	$16 \times 16 \times 8$	-
15	Tconv3	Transposed Convolution	$32 \times 32 \times 16$	weights:$4 \times 4 \times 16 \times 8$ bias:$1 \times 1 \times 16$
16	Relu6	Relu	$32 \times 32 \times 16$	-
17	Conv4	Convolution	$32 \times 32 \times 1$	weights:$3 \times 3 \times 16$ bias:1×1
18	Relu7	Relu	$32 \times 32 \times 1$	-
19	Reg1	Regression	$32 \times 32 \times 1$	-

For VFLA2 the filters are updated via zero padding or truncation based on whether they are being changed from an even size to odd, or vice versa. In case of an increase or decrease in filter size L from even to even or odd to odd, the padding/shrinking factor is calculated as follows:

$$P_{ee/oo}(n) = \frac{|L(n) - L(n-1)|}{2} \quad (6)$$

If the size increases, the filter is padded with $P_{ee/oo}(n)$ additional rows and columns. Similarly, if the size decreases, $P_{ee/oo}(n)$ rows and columns are removed from around the filter. This is depicted in Fig. 3a.

Likewise, when the increase or decrease is from even size to odd, or vice versa, the padding/shrinking factor is calculated as follows:

$$P_{eo/oe}(n) = \frac{|[L(n) - 1] - L(n-1)|}{2} \quad (7)$$

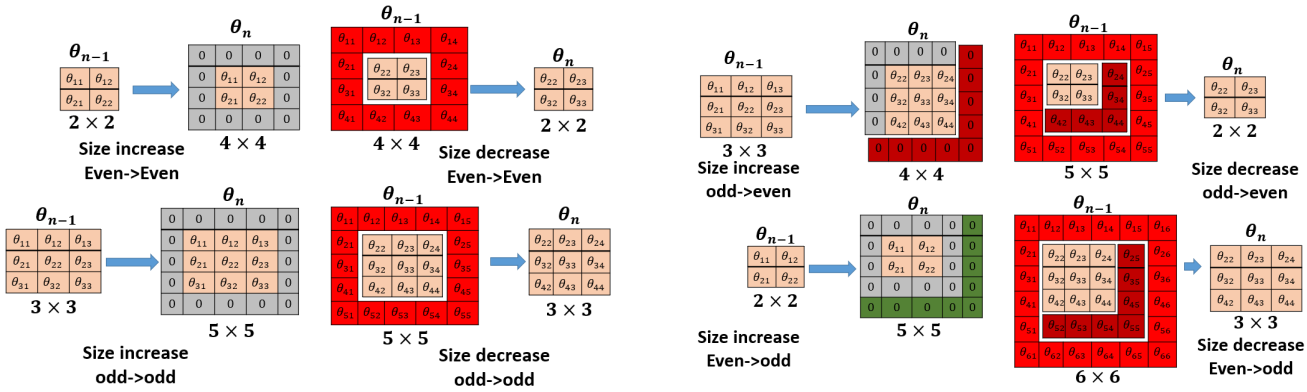


Fig. 3. Size update in VFLA 2. (a) From size Even-Even, Odd-Odd, (b) From size Even-Odd, Odd-Even.

Once the padding/shrinking is carried out, the inherent asymmetry incurred due to changing size from even to odd or vise versa is taken care of by further padding/truncating a column and row pair from the bottom-right of the filter. This is depicted in Fig. 3b, where this additional padding/truncation is shown as dark green and dark red respectively.

Each policy has been applied to 1st and 2nd convolution layer separately. These two policies are not considered for 3rd convolution layer, as the inner-most layer usually synthesizes the features rather than extracts useful features.

B. The OFS-CNN

The OFS-CNN [4] learns each layer's optimum filter size as a hyper-parameter while training. Therefore, each layer's filter size *L(n)* is learned via backpropagation like any other weight or bias.

Each filter size $L(n)$ always remains between two odd numbers, $L_+(n)$ and $L_-(n)$, which are the upper bound and lower bound filter sizes with $L_+(n)$ being the actual size of the filter in question. When the value $L(n)$ exceeds $L_+(n)$ as a result of the training, $L_+(n)$ and $L_-(n)$ are updated to the next odd numbers.

The technique expresses the output of each convolutional layer as a function of the filter size $L_+(n)$, which makes it possible to compute gradients of errors w.r.t the filter size $L(n)$.

III. Simulations and Discussion

A. Datasets

Our work considers three datasets for denoising: the digit dataset [11] comprising of 10 000 synthetic handwritten digits, the Hasy-V2 dataset [13] consisting of over 150,000 instances of handwritten symbols, and the Omniglot dataset [12] containing 1623 different handwritten characters from 50 different alphabets. Each dataset was divided into training, validation and testing datasets with 95%, 2.5% and 2.5% respectively. All images were resized to 32×32 in order to simplify padding concerns. Each minibatch P was set to 500 images per iteration. Each image was normalized to the range $[0, 1]$ to make convergence faster. Each image in the datasets was augmented by randomized rotations $[1°, 90°]$ to reduce overfitting. Salt and pepper noise was added to all images to obtain their noisy versions at the input to each CNN.

B. Experimental setup

The CNN-based denoising autoencoder described in Table I was used as a baseline for both methods. It has several convolution layers in the encoder, followed by several transposed convolution layers in the decoder. Each convolution layer can be described by its spatial dimensions, i.e. length×width×channels×filters. Whereas several convolution layers have multi-channel and multi-filter weights, we limited ourselves with the optimization of spatial dimensions of length and width alone, which we refer to as the *filter size*.

The values for filter sizes in Equation (1) were initialized as 3×3, whereas their values in Equation (2) were initialized as 1×1. λ and f in Equation (3) were set at 1×10^{-3} and 1.5 respectively. The value of ζ in Equation (4) was set at 5 and the value of δ in Equation (5) was set at 2. Finally, The learning rate of the training algorithm was set at 1×10^{-3} and each CNN was trained for 2.5×10^5 iterations.

C. Results

Table III summarizes the learned filter sizes at the end of training for each method, whereas Table II enumerates the final and average MSE values for training and testing. These are evaluated at the regression layer at the end of training. For the baseline, each layer is initialized and kept at the size 3×3 unless otherwise stated in Table II. It is evident that VFLA2 consistently outperformed both the Baseline and the OFS-CNN [4] in terms of its training and testing MSE error giving an average performance improvement of 28.8% from the the minimum average Baseline MSE i.e. 2.77, and 59.8% from OFS-CNN [4]. The white background of the HASY images is the only difference with the other datasets which have darker backgrounds. This may explain the outlier behaviour. For comparison sake, this outlier behavour is experienced not

Fig. 4. Image reconstruction across all data sets

TABLE II
PERFORMANCE COMPARISON. ENTRIES IN BOLD SHOW THE BEST OVERALL PERFORMANCES. ENTRIES IN ITALICS SHOW OUTLIERS

Dataset	Run	Metric	Baseline							VFLA1		VFLA2		OFS-CNN [4]
			3×3 (all layers)	5×5		7×7		9×9						
				conv1	conv2	conv1	conv2	conv1	conv2	conv1	conv2	conv1	conv2	
Digit [11]	Training	Final MSE	1.80	1.86	1.72	1.92	1.15	1.86	**1.01**	1.10	1.17	1.08	1.22	1.48
		Final RMSE	1.34	1.36	1.31	1.38	1.07	1.36	**1.00**	1.04	1.08	1.03	1.10	1.21
	Testing	Average MSE	1.93	4.05	3.76	4.08	4.00	4.04	3.46	1.19	1.29	1.29	**1.18**	1.82
		Average RMSE	1.38	2.01	1.93	2.01	2.00	2.01	1.86	1.09	1.13	1.13	**1.08**	1.35
Omni [12]	Training	Final MSE	3.95	4.20	2.30	2.60	2.05	1.95	2.95	2.45	3.78	**1.45**	2.15	3.20
		Final RMSE	1.98	2.04	1.51	1.61	1.43	1.39	1.71	1.56	1.94	**1.20**	1.46	1.78
	Testing	Average MSE	4.26	*5.12*	2.68	3.02	2.49	2.43	3.45	2.61	4.12	**1.60**	2.28	3.13
		Average RMSE	2.06	2.26	1.63	1.73	1.57	1.55	1.85	1.61	2.03	**1.26**	1.50	1.76
Hasy [13]	Training	Final MSE	*5.40*	*13.2*	*8.90*	*13.0*	*8.20*	**1.78**	*6.65*	*9.50*	*13.5*	*4.70*	*6.30*	*17.77*
		Final RMSE	2.32	3.63	2.98	3.60	2.86	**1.33**	2.57	3.08	3.67	2.16	2.50	4.21
	Testing	Average MSE	*5.31*	*13.50*	*9.02*	*13.39*	*8.69*	*10.38*	*8.10*	*8.86*	*12.19*	***4.63***	*6.21*	*17.12*
		Average RMSE	2.30	3.67	3.00	3.65	2.94	3.22	2.84	2.97	3.49	**2.15**	2.49	4.13
Average			2.83	4.74	3.39	4.33	3.20	2.77	3.12	3.08	4.15	**1.97**	2.45	4.91
Average without outliers			2.32	2.78	2.28	2.56	2.15	2.08	2.26	1.55	2.41	**1.43**	1.68	2.46

only by our proposed methods but also by the other benchmark algorithms. As such, this outlier behaviour does not pose any bias towards any algorithms, since they are all exposed to the same input. If the outliers were removed, all VFLA methods performed better on average than both the Baseline and OFS-CNN [4] with the best being VFLA2 again. In this case, we see improvements of 31.2% and 41.8% from the minimum average without outliers Baseline MSE i.e. 2.08 and OFS-CNN [4] respectively. Furthermore, VFLA was much faster as it took an average of as little as 1800 iterations for convergence, while OFS-CNN [4] took 4×10^4 iterations on average across all datasets. It is also interesting to note that for the Digit Dataset, OFS-CNN [4] learned a filter size of 9×9 that is greater than that of the layer's input feature map of size 8×8. Fig. 4 shows a sample of qualitative results. Due to limit of space, only the VFLA with the lowest MSE is included for each dataset, with the best results highlighted in red.

IV. CONCLUSION

This work has addressed a niche yet often overlooked problem in deep learning. In particular, we have proposed two novel methods to optimise the filter size of a convolutional neural network. We have shown that optimizing one layer rather than all layers can lead to improvement in performance. This is because optimising the weights, the biases, and the filter sizes via backpropagation is a challenging task. Optimising the filter size of only one convolutional layer has also resulted in quicker convergence of our method. Moreover, it has enabled us to circumvent the need to make use of backpropagation, which is well-known to have some shortcomings such as vanishing gradients, resource intensive and lengthy trainings, and sensitivity to noise and other irregularities.

TABLE III
LEARNED FILTER SIZE FOR EACH METHOD. BOLD ENTRIES SHOW THE LEARNED FILTER SIZE L FOR VFLA AND OFS-CNN [4].

Dataset	Layers	Method				
		VFLA1		VFLA2		OFS-CNN [4]
		conv1	conv2	conv1	conv2	
Digit [11]	conv1	**7**	3	**9**	3	**7**
	conv2	3	**7**	3	**7**	**7**
	conv3	3	3	3	3	**9**
	conv4	3	3	3	3	**5**
Omni [12]	conv1	**9**	3	**6**	3	**5**
	conv2	3	**3**	3	**6**	**5**
	conv3	3	3	3	3	**7**
	conv4	3	3	3	3	**3**
Hasy [13]	conv1	**9**	3	**9**	3	**5**
	conv2	3	**5**	3	**6**	**5**
	conv3	3	3	3	3	**7**
	conv4	3	3	3	3	**3**

REFERENCES

[1] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 770-778 (2016)

[2] Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*. (2014)

[3] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. & Others Imagenet large scale visual recognition challenge. *International Journal Of Computer Vision*. **115**, 211-252 (2015)

[4] Han, S., Meng, Z., Li, Z., O'Reilly, J., Cai, J., Wang, X. & Tong, Y. Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition. *2018 IEEE/CVF Conference On Computer Vision And Pattern Recognition*. pp. 5070-5078 (2018)

[5] Zeiler, M. & Fergus, R. Visualizing and understanding convolutional networks. *European Conference On Computer Vision*. pp. 818-833 (2014)

[6] Dhall, A., Ramana Murthy, O., Goecke, R., Joshi, J. & Gedeon, T. Video and image based emotion recognition challenges in the wild: Emotiw 2015. *Proceedings Of The 2015 ACM On International Conference On Multimodal Interaction*. pp. 423-426 (2015)

[7] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Going deeper with convolutions. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 1-9 (2015)

[8] Gong, Y. & Cowan, C. An LMS style variable tap-length algorithm for structure adaptation. *IEEE Transactions On Signal Processing*. **53**, 2400-2407 (2005)

[9] Ibunu, S., Weller, S. & Took, C. A Variable Memory Length Auto Encoder. *2021 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-4 (2021)

[10] Kingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. (2017)

[11] MATLAB Data Sets for Deep Learning. (Available at https://www.mathworks.com/help/deeplearning/ug/data-sets-for-deep-learning.html)

[12] Lake, B., Salakhutdinov, R. & Tenenbaum, J. The Omniglot challenge: a 3-year progress report. *Current Opinion In Behavioral Sciences*. **29** pp. 97-104 (2019)

[13] Thoma, M. The hasyv2 dataset. *ArXiv Preprint ArXiv:1701.08380*. (2017)