Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Self-organized operational neural networks for severe image restoration problems

Junaid Malik[a,*], Serkan Kiranyaz[b], Moncef Gabbouj[a]

[a] *Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland*
[b] *Department of Electrical Engineering, Qatar University, Doha, Qatar*

ARTICLE INFO

Article history:
Received 10 August 2020
Revised and accepted 14 December 2020
Available online 23 December 2020

Keywords:
Image restoration
Image denoising
Artificial neural networks
Operational neural networks

ABSTRACT

Discriminative learning based on convolutional neural networks (CNNs) aims to perform image restoration by learning from training examples of noisy-clean image pairs. It has become the go-to methodology for tackling image restoration and has outperformed the traditional non-local class of methods. However, the top-performing networks are generally composed of many convolutional layers and hundreds of neurons, with trainable parameters in excess of several million. We claim that this is due to the inherently linear nature of convolution-based transformation, which is inadequate for handling severe restoration problems. Recently, a non-linear generalization of CNNs, called the operational neural networks (ONN), has been shown to outperform CNN on AWGN denoising. However, its formulation is burdened by a fixed collection of well-known non-linear operators and an exhaustive search to find the best possible configuration for a given architecture, whose efficacy is further limited by a fixed output layer operator assignment. In this study, we leverage the Taylor series-based function approximation to propose a self-organizing variant of ONNs, Self-ONNs, for image restoration, which synthesizes novel nodal transformations on-the-fly as part of the learning process, thus eliminating the need for redundant training runs for operator search. In addition, it enables a finer level of operator heterogeneity by diversifying individual connections of the receptive fields and weights. We perform a series of extensive ablation experiments across three severe image restoration tasks. Even when a strict equivalence of learnable parameters is imposed, Self-ONNs surpass CNNs by a considerable margin across all problems, improving the generalization performance by up to 3 dB in terms of PSNR.

© 2020 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Image restoration aims at recovering low-level contextual information from noisy and corrupted images. It is one of the key inverse imaging computer vision tasks because the quality of image acquisition is inherently subdued by environmental conditions, quality of the image capturing device, and processes involved in obtaining a digital image from photosensors. The earliest works were based on the basic assumption that smoothing by averaging leads to denoising (Alvarez, Lions, & Morel, 1992; Donoho, 1995; Smith & Brady, 1997). Such methods were then surpassed by the non-local class of methods (Buades, Coll, & Morel, 2005; Mahmoudi & Sapiro, 2005; Mairal, Bach, Ponce, Sapiro, & Zisserman, 2009), among which the method BM3D (Dabov, Foi, Katkovnik, & Egiazarian, 2007) is widely considered to be state-of-the-art. In recent years, the focus has shifted towards *learning* the restoration problem, by posing it in a discriminative paradigm of training on noisy-clean image pairs. Convolutional Neural Networks (CNNs)-based approaches have rapidly reached apex performance in almost all learning-based computer vision problems (He, Zhang, Ren, & Sun, 2016; Krizhevsky, Sutskever, & Hinton, 2012; Shelhamer, Long, & Darrell, 2017), and image restoration is no exception (Lempitsky, Vedaldi, & Ulyanov, 2018; Zhang, Zuo, Chen, Meng & Zhang, 2017; Zhang, Zuo, Gu & Zhang, 2017).

CNNs are artificial neural networks composed of stacked layers of convolutional neurons, each filtering local receptive fields within the input feature maps by convolving them with learnable filter banks. Their weight sharing and local connections significantly reduce the size of parameter space as compared to multilayer perceptrons (MLPs) and make them especially efficient for large grid-structured data such as images. While the earlier CNN models were not particularly deep (LeCun, Bottou, Bengio, & Haffner, 1998), faster implementations on GPU (Cires, Meier, Masci, & Gambardella, 2003) ushered in an era of deeper and more complex architectures (Dabov et al., 2007; Mahmoudi & Sapiro, 2005). Contemporary CNN architectures are generally composed of tens of layers and their performance is generally observed to be correlated with their *depth* (number of layers)

* Correspondence to: Korkeakoulunkatu 7, 33720, Tampere, Finland.
E-mail address: junaid.malik@tuni.fi (J. Malik).

https://doi.org/10.1016/j.neunet.2020.12.014
0893-6080/© 2020 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

(Conneau, Schwenk, Le Cun, & Barrault, 2017). Some of the state-of-the-art CNNs for image restoration consist of learnable parameters in the order of millions (Zou, Lan, Zhong, Liu, & Luo, 2019), despite dealing with mild restoration problems where the corrupted images still retain most of the semantic information. Moreover, a significant amount of training resources is required to train deep architectures in order to avoid overfitting and ensure proper generalization.

Despite their wide-scale adoption, as identified in Kiranyaz, Ince, Iosifidis and Gabbouj (2020), Kiranyaz, Malik et al. (2020), the necessity of deeper and wider CNN architectures stems from some of the inherent drawbacks in the convolutional model. Firstly, the convolutional neuron model implies a strict linear transformation, where the only source of non-linearity may stem from the point-wise non-linear activation. Therefore, a very high number of neurons with interwoven non-linear activations is required in order to synthesize a rich enough hypothesis for challenging restoration problems. A remedy to this has been recently proposed in Kiranyaz, Ince et al. (2020), Kiranyaz, Malik et al. (2020) using the so-called Operational Neural Networks (ONNs) which embed non-linear operations inside the patch-wise transformations as an alternative to the linear convolutional model. Specifically, the linear transformation of weight multiplications and additions (forming the convolution operation) are generalized to non-linear mappings, called nodal and pool functions, respectively. These nonlinear mappings are defined in an operator set library and searched for each learning problem individually. Similar to their predecessors, Generalized Operational Perceptrons (GOPs), which were shown to be superior to traditional Multi-Layer Perceptrons (MLPs) (Kiranyaz, Ince, Iosifidis, & Gabbouj, 2017a, 2017b; Tran, Kiranyaz, Gabbouj, & Iosifidis, 2020a, 2020b), ONNs were shown to outperform equivalent and even deeper CNN architectures across a variety of computer vision problems, including AWGN image denoising (Kiranyaz, Ince et al., 2020; Kiranyaz, Malik et al., 2020).

In ONNs, the choices for nodal and pool operators are critical towards generating an optimal degree of non-linearity for a given problem. While their generic formulation enables the flexibility to incorporate any non-linear transformation, their efficacy relies on two key factors; (i) curation of a diverse enough operator set library and (ii) the selection of the optimal operators for the problem at hand. In the case of the former, it is possible that the optimal non-linear transformation required for the restoration problem cannot be expressed by a well-known function such as a sinusoid or an exponential function. Therefore, there always stands a possibility that the operator set library, no matter how large it is, remains insufficient. Secondly, the search paradigms used in Kiranyaz, Ince et al. (2020) and Kiranyaz, Malik et al. (2020); Greedy Iterative Search (GIS) and Synaptic Plasticity Monitoring (SPM) respectively, both require additional training runs for each learning problem, in order to converge towards an optimal set of operators per hidden layer. Therefore, using ONNs for the image restoration problem would imply selecting an appropriate operator set library and searching over it separately for each of the noise characteristics being studied. This makes their usage cumbersome and perhaps impractical for large datasets. Moreover, even if an ideal convergence is assumed, GIS-based operator search only yields a homogeneous configuration of layers i.e. all neurons in a given layer are limited to having the same non-linear transformation. Extending it for a heterogeneous configuration, where each neuron has a distinct operator set, would make the search prohibitively expensive. The SPM method proposed in Kiranyaz, Malik et al. (2020) remedies this, but it relies on a suitable initial random assignment of operators to neurons and makes a strong assumption that the choice of the operator of a neuron in a layer does not affect the other neurons of the current layer and the neurons in the previous layer.

In order to solve severe and diverse restoration problems, we identify the need for a self-reliant alternative to convolutional neurons for embedding non-linearities, without the need for additional training runs, as in ONNs. To accomplish this objective in this study, we make the following contributions:

– we propose, Self-ONN, a self-organized variant to the ONN which does not require prior curation of an operator set library and consequently, the need to search over it. Self-ONNs with generative neurons can leverage the Taylor series-based approximation to generate "any" nodal operator, which is optimized as part of the learning process, thus voiding the need for any prior training runs.
– we compare the proposed approach with equivalent CNN and ONN architectures having the same number of learning units (neurons) and network parameters. The experiments are conducted on severely corrupted images and tested on five benchmark datasets and three different noise types.

An extensive set of experimental results demonstrates that the proposed Self-ONNs exhibit a superior generalization and training performance compared to both equivalent ONNs and CNNs across all image restoration problems. The rest of the paper is structured as follows: In Section 2, we briefly describe some of the contemporary methods employed for image restoration. Section 3 provides technical details of Self-ONNs by comparing their formulation to vanilla ONNs and CNNs. Section 4 elucidates the experimental setup and the characteristics of networks and datasets used in this study. Section 5 provides key insights into the results and findings of the study, while Section 6 concludes the paper and suggests possible future research directions.

2. Related work

Prior to the advent of CNN-based discriminative learning, the non-local class of methods dominated the field of image restoration (Buades et al., 2005; Mahmoudi & Sapiro, 2005; Mairal et al., 2009). Arguably, the most successful technique belonging to this class is that of BM3D (Dabov et al., 2007). The method works by creating a pool of similar non-local patches across the image. A collaborative filtering procedure is subsequently applied where the pools of patches are projected to a higher-dimensional space and denoised by shrinking the transform coefficients. Recently, CNNs have become the *de-facto* standard. Earlier, in Burger, Schuler, and Harmeling (2012), it was observed that an MLP can be trained to competitively reproduce the results of BM3D. In Zhang, Zuo, Chen et al. (2017), a deep CNN architecture was proposed that successfully applied batch normalization and residual learning principles to achieve competitive results for various degrees of AWGN. The study in Zhang, Zuo, and Zhang (2018) aimed to incorporate spatial-invariance by proposing to supplement the inputs with an additional noise map so that the CNN can learn spatially-invariant encodings for denoising. In Tian, Xu, and Zuo (2020), the residual framework adopted in Zhang, Zuo, Chen et al. (2017) is used and extended with batch renormalization and dilated convolutions to address the problems with small mini-batch and limited receptive fields, respectively. In Zhang, Zuo, Gu et al. (2017), the discriminative learning paradigm is integrated as a modular part of model-based optimization by utilizing the variable-splitting technique. The model architecture employed is similar to that used in Zhang, Zuo, Chen et al. (2017). The authors of Zou et al. (2019) employed a very deep network, composed of 52 layers, with a global and a local residual framework to tackle high-level AWGN denoising. Generally, the proposed CNN-based methods employ considerably deep architectures and learn from large-scale datasets; consisting of training examples in the order of 10^5. Moreover, the noise characteristics of input images are generally mild and preserve the contextual information of the image quite well. Therefore, there is a need to evaluate learning models for restoring highly corrupted images having diverse and severe noise characteristics. Furthermore, as of now, there exists scarce literature that explores the possibility of exploiting non-linearity in the realm of CNNs for image denoising (Kiranyaz, Ince et al., 2020; Wang, Yang, Xie, & Yuan, 2019; Zoumpourlis, Doumanoglou, Vretos, & Daras, 2017). While ONNs provide a viable alternative, the need of defining a fixed operator set library in advance and performing an exhaustive search to find the best operators may render their usage impractical, especially for large-scale datasets.

3. Self-organized operational neural networks

In ONNs, the primary building block is the operational neuron model which extends the principles of GOPs to the convolutional realm. While retaining the favorable characteristics of sparse-connectivity and weight-sharing in a CNN, an ONN provides the flexibility to incorporate *any* non-linear transformation within local receptive fields without the overhead of additional trainable parameters. In this section, we provide a brief overview of the convolutional operation in CNN and explain how ONNs generalize it.

3.1. Operational neural networks

In a convolutional neuron, given the output of layer $l-1$, the pre-activation output of the kth convolutional neuron in layer l is calculated as:

$$x_l^k(i,j) = \sum_{u=0}^{m-1}\sum_{v=0}^{n-1} w_l^k(u,v)\, y_{l-1}(i-u, j-v) \tag{1}$$

where $y_{l-1} \in \mathbb{R}^{M\times N}$ and the weight kernel $w_l^k \in \mathbb{R}^{m\times n}$. For the sake of brevity, unit stride and dilation are assumed, and the input is padded with zeros before the convolution operation in order to preserve the spatial dimensions. An alternate formulation of the operation of (1) is now presented. Firstly, y is reshuffled such that values inside each $m \times n$ sliding block of y_{l-1} is vectorized and concatenated as rows to form a matrix $Y_{l-1} \in \mathbb{R}^{\hat{M}\times\hat{N}}$ where $\hat{M} = MN$ and $\hat{N} = mn$. This operation is commonly referred to as "*im2col*" and is critical in conventional GEMM-based convolution implementations (Chetlur et al., 2014). Secondly, we construct a matrix $W_l^k \in \mathbb{R}^{\hat{M}\times\hat{N}}$ whose rows are repeated copies of $\overrightarrow{w_l^k} = vec\left(W_l^k\right) \in \mathbb{R}^{mn}$, where $vec(\cdot)$ is the vectorization operator. Each element of W_l^k is given by the following equation:

$$W_l^k(i,j) = \overrightarrow{w_l^k}(i) \tag{2}$$

The convolution operation in (1) can then be represented as follows:

$$x_l^k = vec_{M\times N}^{-1}\left(\sum_j \left(Y_{l-1} \otimes W_l^k\right)\right) \tag{3}$$

where $\otimes$ represents the Hadamard product, $\sum_j$ is the summation across jth dimension. In (3), $vec_{M\times N}^{-1}$ is the inverse vectorization operation that reshapes back to $M \times N$. The formulation given in (3) can now be generically reformulated to represent the forward-propagation through an operational neuron:

$$x_l^k = vec_{M\times N}^{-1}\left(\phi_l^k\left(\psi_l^k\left(Y_{l-1}, W_l^k\right)\right)\right) \tag{4}$$

where $\psi(\cdot): \mathbb{R}^{M\times N} \to \mathbb{R}^{M\times N}$ and $\phi(\cdot): \mathbb{R}^{\hat{M}\times\hat{N}} \to \mathbb{R}^{\hat{M}}$ are termed as *nodal* and *pool* functions, respectively. Finally, after applying the activation function f_l^k, we get the output of the neuron:

$$y_l^k = f_l^k\left(vec_{M\times N}^{-1}\left(\phi_l^k\left(\psi_l^k\left(Y_{l-1}, W_l^k\right)\right)\right)\right) \tag{5}$$

Given an operator set; a triplet of $\left(\psi_l^k, \phi_l^k, f_l^k\right)$, an operational neuron implements the formulation given in (5). It can be noticed here that the convolutional neuron is a special case of an operational neuron with nodal function $\psi(\alpha,\beta) = \alpha * \beta$ and pooling function $\phi(\cdot) = \sum_i \cdot$

3.2. Self-organized operational neural networks

The Taylor series expansion of an infinitely differentiable function $f(x)$ about a point a is given as:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{6}$$

The Qth order truncated approximation, formally known as the Taylor polynomial, takes the form of the following finite summation:

$$f(x)^{(Q,a)} = \sum_{n=0}^{Q} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{7}$$

The above formulation can approximate any function $f(x)$ sufficiently well in the close vicinity of a. With the use of an activation function that bounds the neuron's input feature maps within $[a-\gamma, a+\gamma]$, the formulation of (7) can be exploited to form a composite nodal operator where the coefficients of the powers of x are the learned parameters of the network. Following the same notation as introduced in Section 3.1, the nodal operator of the kth generative neuron in the lth layer would take the following general form:

$$\psi_l^k\left(Y_{l-1}, \boldsymbol{W_l^k}, Q, a\right) = \sum_{q=1}^{Q} Y_{l-1}^q \otimes W_l^{k(q)} \tag{8}$$

where $\boldsymbol{W_l^k} \in \mathbb{R}^{\hat{M}\times\hat{N}\times Q}$ is the three-dimensional weight matrix and $W_l^{k(q)} \in \mathbb{R}^{\hat{M}\times\hat{N}}$ is the qth slice of $\boldsymbol{W_l^k}$. The 0th order term $\boldsymbol{a}$, the DC bias, is ignored as its additive effect can be compensated by the learnable bias parameter of the neuron. Back-propagation (BP) through this nodal operator is now trivial to accomplish. Eqs. (9) and (10) provide the derivatives with respect to the input Y_{l-1} and the qth slice weights, $W_l^{k(q)}$, respectively:

$$\frac{d\psi_l^k}{dY_{l-1}} = \sum_{q=1}^{Q} qY_{l-1}^{q-1} \otimes W_l^{k(q)} \tag{9}$$

$$\frac{d\psi_l^k}{dW_l^{k(q)}} = Y_{l-1}^q \tag{10}$$

During the learning process, as the weights are updated by BP, this formulation enables the network to spawn novel nodal transformations that are optimized to achieve the given learning objectives. Moreover, the resulting polynomials are functions that provide accurate approximation only within the operating range $[a-\gamma, a+\gamma]$, which makes them easier to compute as opposed to functions that generally comprise the operator set library of an operational neural network (Kiranyaz, Ince et al., 2020). In addition, the formulation is heterogeneous by design and enables different nodal transformations for different neurons in a layer. The heterogeneity offered by such a formulation is incorporated in all neurons of all the layers, including the output layer. This provides a crucial advantage over ONNs, where the output layer operators are always confined a priori to a fixed operator set (Kiranyaz, Ince et al., 2020). Expanding on the intuition that earlier layers of neural networks are involved with feature extraction while the latter ones deal with classification, we can see that self-organizing operational neurons not only can extract

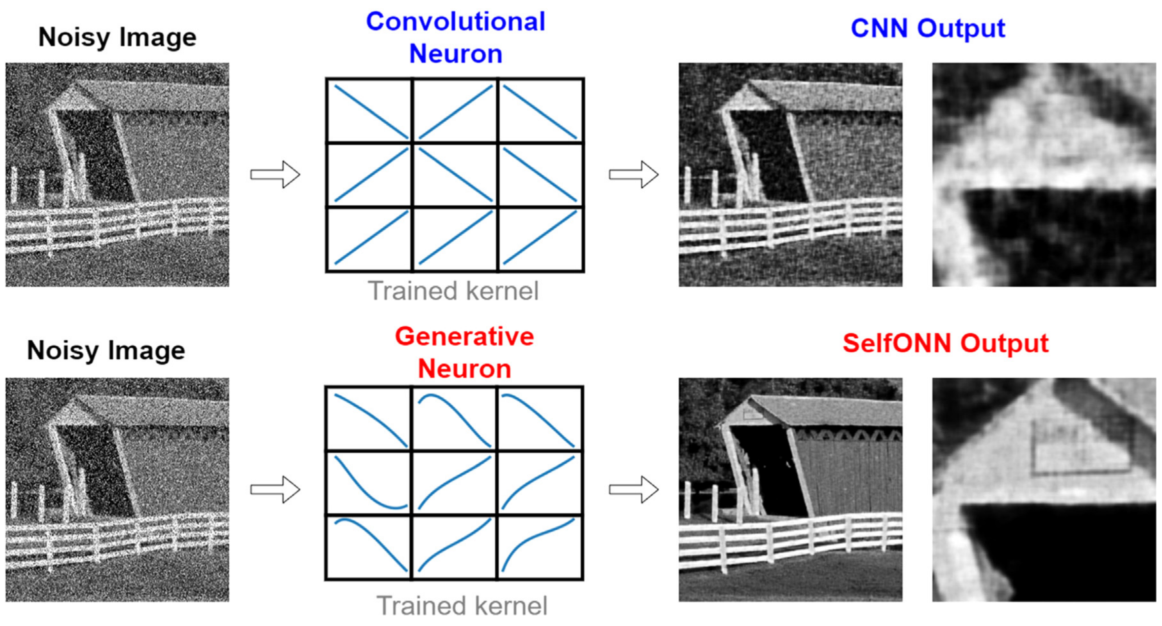

Fig. 1. An illustration of the 3 × 3 trained kernels of the convolutional versus generative neurons. The linear transformations of the convolutional neuron fail to restore key image contents, whereas the non-linear transformations posed by the generative neuron enable significantly superior restoration performance.

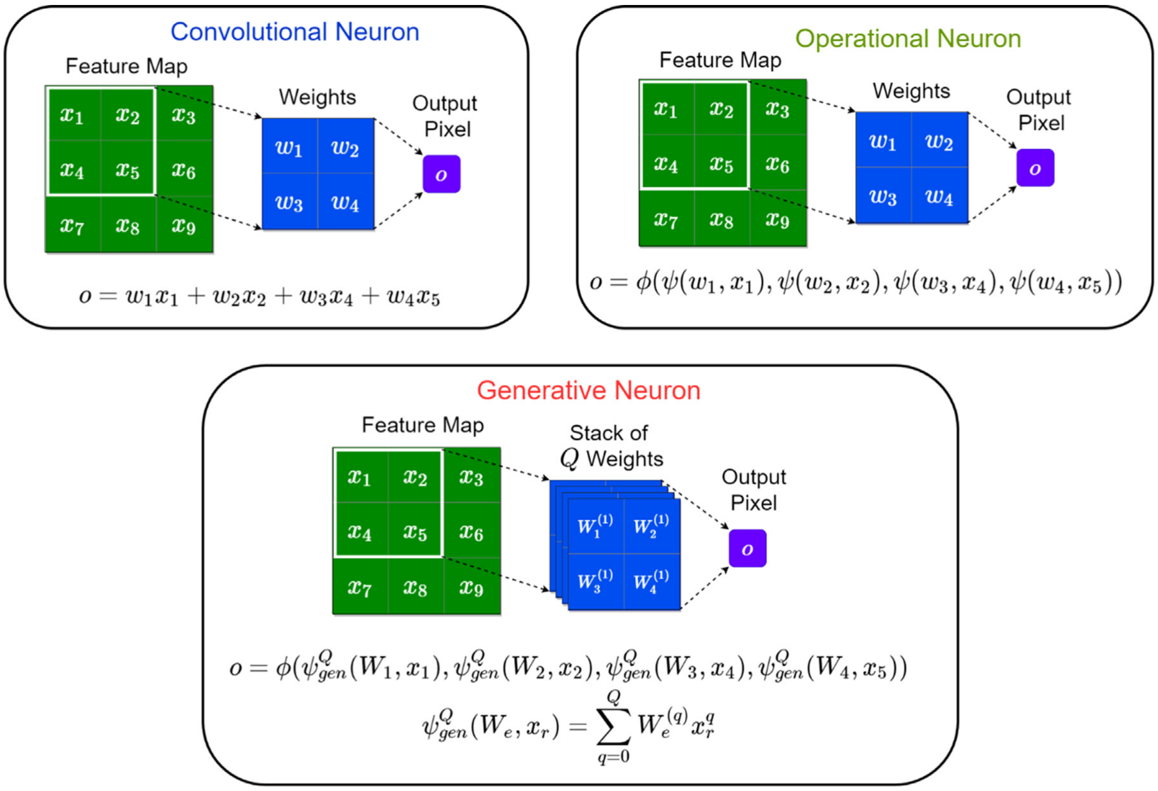
Convolutional Neuron
Feature Map
Weights
Output Pixel
$o = w_1x_1 + w_2x_2 + w_3x_4 + w_4x_5$
Operational Neuron
Feature Map
Weights
Output Pixel
$o = \phi(\psi(w_1, x_1), \psi(w_2, x_2), \psi(w_3, x_4), \psi(w_4, x_5))$
Generative Neuron
Feature Map
Stack of Q Weights
Output Pixel
$o = \phi(\psi^Q_{gen}(W_1, x_1), \psi^Q_{gen}(W_2, x_2), \psi^Q_{gen}(W_3, x_4), \psi^Q_{gen}(W_4, x_5))$
$\psi^Q_{gen}(W_e, x_r) = \sum_{q=0}^{Q} W_e^{(q)} x_r^q$

Fig. 2. Illustration of the formulations of the convolutional, operational, and self-organized (generative) operational neuron.

more discriminative features but can also generate better decision boundaries, by virtue of the optimized nodal transformations. Furthermore, as depicted in Fig. 2, the set of Q-weights corresponding to each element of the $K_x \times K_y$ receptive field is unique, and consequently, corresponds to different transformations. This provides an unprecedented level of diversity where all the connections between individual elements of the receptive field to the corresponding weights are governed by different transformation functions (i.e., the nodal operators for ONNs), as shown in Fig. 1. Such a level of heterogeneity is not possible in the formulation of ONNs since a single nodal operator is assigned to each neuron and thus is used by *all* its connections to the previous layer neurons with distinct weights (parameters). Therefore, the proposed generative neuron is not simply another nodal transformation, but it adds another layer of generalization on top of the operational neuron idea by enabling the fore-mentioned heterogeneity.

Finally, and most importantly, the property of generating nodal transformations on-the-fly alleviates the need for prior training runs to search for an appropriate operator set, which is a practical challenge in the case of ONNs. Therefore, an operational neuron equipped with this nodal operator termed as a *generative neuron*, and the resulting network configuration, the Self-ONN, provides a plug-and-play replacement for the convolution-based models and, therefore, can directly be integrated into any contemporary CNN architecture.

3.3. Relationship to conventional CNNs

CNNs are indeed a subset of ONNs, corresponding to nodal and pool functions of multiplication and summation, respectively. Similarly, a convolutional neuron is also a special case of a generative neuron with $Q = 1$. Moreover, if the pooling operator ϕ is fixed to summation the formulation of (8) can be interpreted as Q independent convolutions. The forward propagation through the neuron can then be achieved using a single large convolution operation. This property enables fast implementation of generative neurons and provides a significant computational edge when compared to the operational ones.

4. Experiments

4.1. Training and network constraints

Convolutional neural networks' efficacy is related directly to their depth. But deeper networks require more data and backpropagation iterations to learn efficiently. Therefore, generalization performance comparison alone is not sufficient for measuring the true learning ability of a neuronal model because large-scale data coupled with increasing model complexity can potentially mask any shortcomings of the neurons. A more useful comparison entails limited availability of training resources such as the number of labeled samples, training iterations, and model complexity To accomplish this, we apply the following training constraints in our experiments:

(i) Training data is kept scarce. Only 10% (100) patches are used to train each model, while the remaining 90% (900) patches, as well as high-resolution noisy images from BSD12 and BSD68 datasets, are held out for testing. It is worth noting here that training on such a small portion of data is quite uncommon for deep CNNs, where the ratio of training samples to test samples is always greater than 1 (Zhang, Zuo, Chen et al., 2017).
(ii) The number of epochs is limited to 100.
(iii) Model architectures are compact; limited to two hidden layers.
(iv) Stochastic Gradient Descent (SGD) with momentum-based optimization is used in BP. Adaptive optimizers such as Adam (Kingma & Ba, 2015), and RMSProp (Tieleman & Hinton, 2012), are omitted.
(v) In all noise types, the degree of corruption is severe (SNR < 0 dB) resulting in the loss of nearly all the contextual information.

All the networks are trained with a learning rate of 0.01, a momentum of 0.9, and a mini-batch size of 4, using 10-fold cross-validation on the training set. For each experiment, three different randomly-initialized training runs are performed and the best-performing run with respect to the training loss is chosen.

4.2. Noise models and datasets

We employ three types of image corruption models; (i) additive white Gaussian noise (AWGN), (ii) impulse noise, and (iii) speckle noise over 1000 randomly chosen images from the PASCAL dataset (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010). For each noise model, we further employ two different levels of distortion. All images are converted to grayscale and resized as 60×60 patches. We perform 10-fold cross-validation and compute the test performance as the average across all folds of the held-out test set. Specifically, in each fold, the network is trained on 100 patches and tested on the remaining 900. In addition, we also evaluate the restoration performance of the trained model on noisy high-resolution grayscale images from BSD12, BSD68, Kodak24, and McMaster (Wu, 2011) datasets. In the case of AWGN, we corrupt each of the images to have −5 dB and −2 dB SNR levels (defined as the ratio of the image variance to the mean square error of the noise). This is equivalent to average σ values of 90 and 60 respectively of the added Gaussian noise. For impulse noise denoising, a fixed-value impulse noise with a probability of 0.4 and 0.2 is applied; consequently replacing 40% and 20% of the pixels with either the darkest or brightest pixel values possible within the data range, respectively. For speckle noise, we employ the model used in Bioucas-Dias and Figueiredo (2010) where the noise probability is given by the Gamma distribution:

$$p(n) = \frac{M}{\Gamma(M)} e^{-nM} n^{M-1} \tag{11}$$

For our experiments, corrupted images corresponding to acute noise levels of $M = 5$ and M=10 are used. Fig. 3 provides examples of some images corrupted with the various kinds and levels of noises used in this study. For all the problems, we minimize the mean-squared loss and use the Peak Signal-to-Noise Ratio (PSNR) as the performance metric:

$$PSNR\left(x_{orig}, x_{noisy}\right) = 10 \log_{10}\left(\frac{MAX_{x_{orig}}^2}{\sum_N \left(x_{orig} - x_{noisy}\right)^2}\right) \tag{12}$$

where $MAX_{x_{orig}}$ is the maximum possible peak of x_{orig} in the data range.

4.3. Network architecture

We use a compact network architecture comprising of 2 hidden layers, as depicted in Fig. 4. Moreover, as presented in Section 3.2, a generative neuron has $\boldsymbol{Q}$ times more network parameters compared to a convolutional neuron. Therefore, in order to investigate the exact impact of employing these additional learnable parameters, we conduct a series of ablation tests. Specifically, we compare a Self-ONN with the order $\boldsymbol{Q} > 1$ to the corresponding CNN with the same number of neurons, as well as with the CNNs having roughly the same number of network parameters. This equivalence is reached by increasing the width of each convolutional layer until the number of network parameters is similar. The exact number of neurons in each hidden layer is provided in Table 1.

We propose three Self-ONN configurations; Self-ONN-3, Self-ONN-5 and Self-ONN-7 corresponding to $\boldsymbol{Q} \in [3, 5, 7]$, respectively. A higher value of Q enables greater non-linearity, but at the expense of more trainable parameters. Therefore we experiment with different Q values in order to empirically analyze this trade-off for the denoising problem. The number of generative neurons in the first and 2nd hidden layers is set to 6 and 10, respectively. The corresponding CNNs with the same number of parameters are termed CNN-3, CNN-5, and CNN-7, while the CNN having the same number of neurons as the Self-ONN is referred to as CNN-1 in the study.

For Self-ONNs, with the use of **tanh** as activation, the value of $\boldsymbol{a}$ is naturally set to 0 in (8). Additionally, we restrict the choice of pooling function $\boldsymbol{\phi}$ to summation. Table 1 provides the

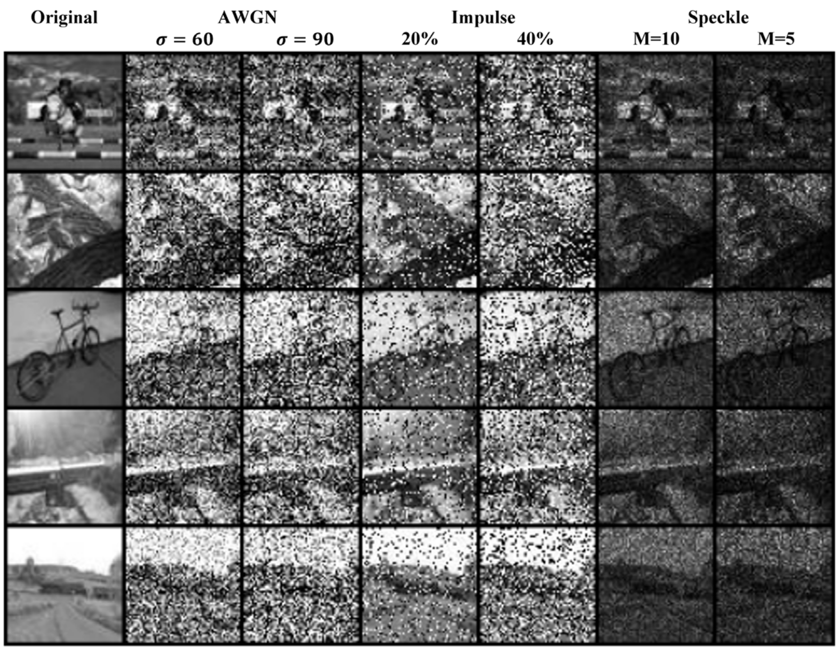

Fig. 3. Examples of target patches and their corrupted counterparts for each type of noise.

Table 1
Architectural details for the different networks.

Network	Neurons in Layer 1	Neurons in Layer 2	Total neurons	Parameters (k)	Kernel sizes
Self-ONN-7	6	10	16	26.3	11,7,3
Self-ONN-5	6	10	16	18.8	11,7,3
Self-ONN-3	6	10	16	11.3	11,7,3
CNN-1	6	10	16	3.7	11,7,3
CNN-3	11	18	29	11.2	11,7,3
CNN-5	14	24	38	18.4	11,7,3
CNN-7	18	27	45	26.3	11,7,3
DnCNN (Zhang, Zuo, Chen et al., 2017)	–	–	1024	500	3

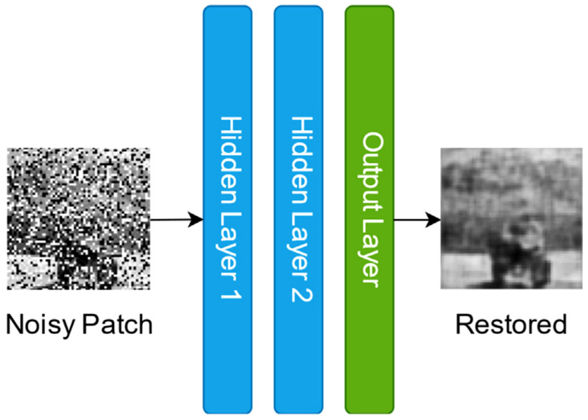

Fig. 4. The network architecture employed in this study.

architectural details of each network. We also compare each Self-ONN to an equivalent ONN for AWGN. Moreover, a comparison with a deep CNN, albeit unfair, is also provided. We use the 17-layer architecture proposed in Zhang, Zuo, Chen et al. (2017) which has 64 neurons in each layer.

5. Results and discussion

Table 2 shows the restoration performance in terms of PSNR over the patches in the test set of Pascal, and over the high-resolution images from BSD12, BSD68, Kodak, and McMaster datasets for all the three restoration problems. Fig. 5 depicts the training curves for the three problems and all the networks included in this study, while Fig. 8 provides visual examples of the denoising performance of various networks.

5.1. Self-ONNs versus equivalent CNNs

We first aim to investigate the performance of generative neurons and convolutional neurons when treated as standalone learning units. To accomplish this, we compare Self-ONNs against CNN with the same number of neurons (CNN-1) for all problems and datasets. We observe from Table 2 that Self-ONNs consistently achieve on average, 1.03 dB, 3.68 dB, and 1.06 dB PSNR improvement in the generalization performances across the different levels of AWGN, impulse, and speckle noise models respectively. The performance gap is exceptionally high for the case of impulse noise across all three datasets. Moreover, even for the case of AWGN where the convolutional neurons are expected to fare better, we observe a clear gap in both training and generalization performances consistently across the three datasets. Furthermore, the performance gain with generative neurons seems to be correlated with the order, Q, as higher order Self-ONN (e.g., Self-ONN-7 and Self-ONN-5) achieves better performance when compared to the lower order variant (e.g., Self-ONN-3).

Fig. 6 shows the comparison of the generalization performance across all datasets and problems, between each Self-ONN configuration and its equivalent CNN, having the same number of network parameters. We can observe a clear trend where even

Table 2
PSNR performances for all models across the three noise types over the test sets.

Noise	Params	Dataset	Self-ONN			CNN				Deep CNN
			Self-ONN-7	Self-ONN-5	Self-ONN-3	CNN	CNN-3	CNN-5	CNN-7	DnCNN
AWGN	$\sigma = 90$	Set12	**21.29**	21.14	21.14	20.76	20.87	20.88	20.85	17.80
		Set68	**20.39**	20.22	20.19	19.20	19.25	19.28	19.30	17.48
		Pascal	**19.47**	19.41	19.37	18.74	18.80	18.81	18.80	16.75
		Kodak	**22.39**	22.23	22.08	21.02	21.17	21.20	21.22	18.67
		McM	21.71	**21.87**	21.77	20.23	20.26	20.19	20.14	17.90
	$\sigma = 60$	Set12	22.37	22.36	**22.38**	21.83	21.97	21.95	21.93	19.84
		Set68	**21.80**	21.73	21.64	20.31	20.43	20.40	20.40	19.46
		Pascal	**20.56**	20.54	20.53	19.77	19.85	19.86	19.86	18.55
		Kodak	**23.17**	23.16	23.09	22.11	22.33	22.28	22.29	20.59
		McM	22.73	22.88	**22.96**	21.26	21.11	21.19	21.18	20.26
IMPULSE	40%	Set12	**23.35**	23.34	23.30	20.39	20.45	20.52	20.46	19.15
		Set68	22.30	**22.36**	22.22	19.42	19.47	19.50	19.47	18.61
		Pascal	21.50	**21.53**	21.47	18.45	18.52	18.53	18.52	17.82
		Kodak	**25.03**	24.91	24.87	20.66	20.77	20.80	20.76	19.58
		McM	**25.20**	25.01	25.17	20.22	20.44	20.37	20.33	19.64
	20%	Set12	24.73	**24.84**	24.66	21.87	22.08	21.98	22.01	22.78
		Set68	24.05	**24.11**	23.88	20.70	20.83	20.81	20.78	22.04
		Pascal	23.11	**23.17**	22.92	19.88	19.98	19.98	19.98	21.03
		Kodak	26.45	**26.61**	26.05	21.89	22.11	22.05	22.06	24.13
		McM	26.29	**26.58**	26.16	21.43	21.60	21.60	21.60	23.63
SPECKLE	M=5	Set12	**19.18**	18.91	18.86	17.92	18.02	17.68	17.94	17.80
		Set68	**18.72**	18.54	18.56	17.86	17.91	17.69	17.89	17.64
		Pascal	19.20	19.18	**19.21**	18.76	18.75	18.71	18.75	17.97
		Kodak	**18.76**	18.54	18.37	17.12	17.16	16.92	17.14	17.16
		McM	**21.34**	21.11	21.08	19.08	19.02	18.79	19.05	19.50
	M=10	Set12	**20.32**	20.25	20.10	19.75	19.65	19.65	19.69	19.12
		Set68	**19.30**	19.21	19.16	18.54	18.52	18.57	18.65	18.45
		Pascal	20.51	**20.52**	20.49	19.81	19.84	19.84	19.87	19.22
		Kodak	**19.59**	19.42	19.32	18.30	18.20	18.30	18.42	18.40
		McM	22.06	**22.10**	21.93	20.18	20.18	20.14	20.22	20.70

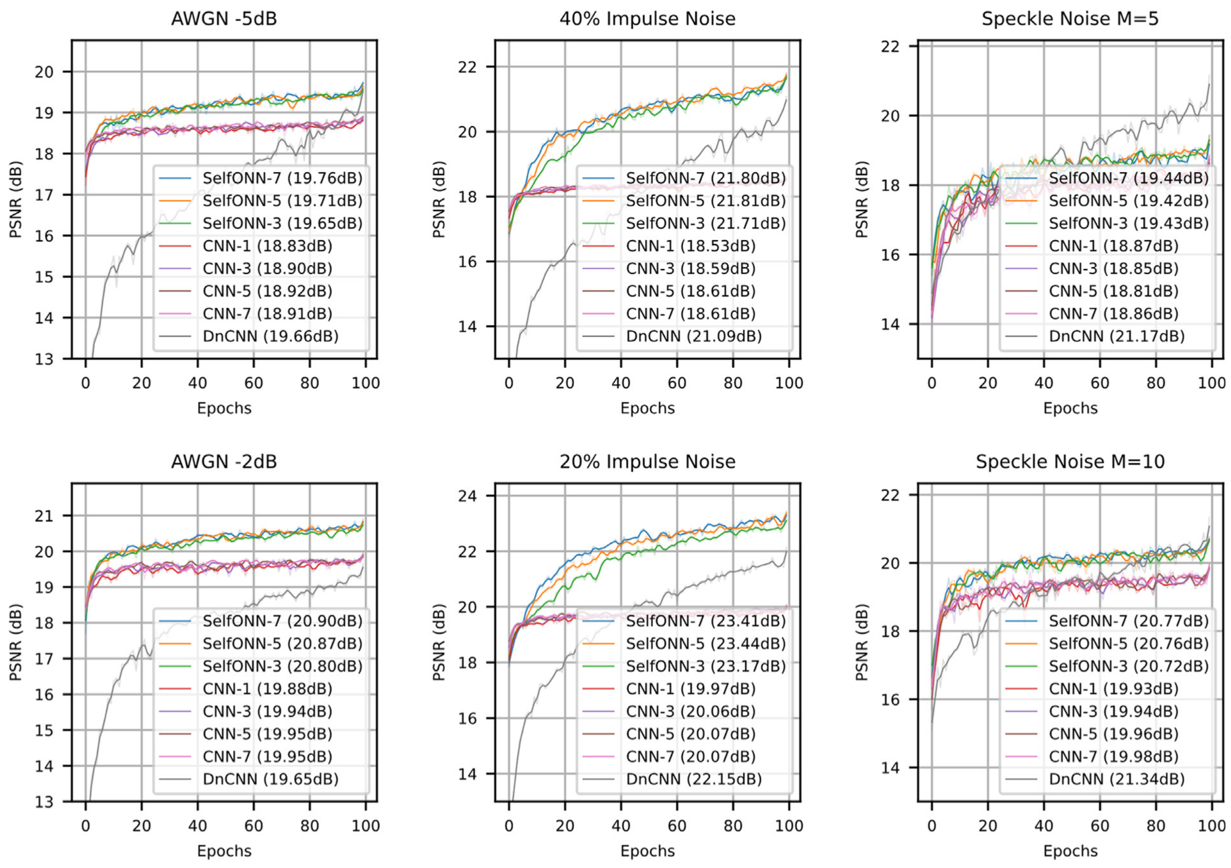

Fig. 5. PSNR curves versus BP epochs for images restored from different noise models of all the networks. The peak value is provided inside the parenthesis.

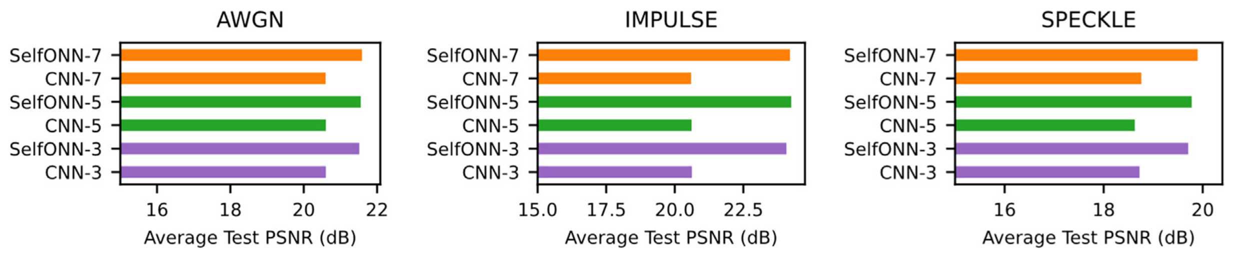

Fig. 6. PSNR performances averaged across the three datasets for Self-ONNs with order $Q = 3$, 5, and 7 as compared to equivalent CNNs over the test set. Colored bars with the same color correspond to networks with the same number of network parameters.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Self-ONN and ONN PSNR performances on the restoration of images corrupted with AWGN of $\sigma = 90$.

	BSD12	BSD68	Pascal	Kodak	McMaster
Self-ONN-7	**21.29**	**20.39**	**19.47**	**22.39**	**21.71**
Self-ONN-5	21.14	20.22	19.41	22.23	21.87
Self-ONN-3	21.14	20.19	19.37	22.08	21.77
ONN	20.91	19.92	18.85	20.97	20.48

a lower order Self-ONN such as the one with $Q = 3$, significantly outperforms its convolutional counterpart (more than 1 dB PSNR improvement). This validates our claim that adding more parameters in the form of wider convolutional layers with more neurons is sub-optimal. In other words, instead of simply increasing the number of neurons, the generative neurons of Self-ONNs with optimized non-linear nodal operators achieve better diversity and thus have a higher impact on the restoration performance.

5.2. Self-ONNs versus equivalent ONNs

In order to gauge the performance of Self-ONNs against equivalent size ONNs, we configured an ONN with the same network architecture as in Fig. 4 using the operator combinations provided in Kiranyaz, Ince et al. (2020) for AWGN noise with σ set to 90. Table 3 provides the comparison over the test set of the resulting PSNR for Self-ONNs and ONNs across the three datasets.

All Self-ONN variants outperform the ONN on this task across all three datasets. These results suggest that the nodal operators selected within the operator set library of ONNs, although better than the sole convolution operator of CNNs, may still not be optimal and the synthesized nodal operators in Self-ONNs provide a much better alternative.

5.3. Self-ONNs versus deep CNNs

We can see from Fig. 5 that the deep 17-layer CNN performs well on the training set and consistently outperforms the compact 2-layer CNNs across all noise categories. On speckle noise with M=5 and M=10, it achieves 21.17 dB and 21.34 dB respectively on the training set which is the top among all competing architectures, even surpassing the Self-ONN variants. However, as is evident from the test set performance shown in Table 2, the model severely overfits the training data and fails to generalize well. In the case of AWGN and impulse noise, the test set performance of the deep model is even worse than the shallow CNNs, while on speckle noise it is barely at par with them. This is an important observation that validates the notion that deeper models unless coupled with ample training resources, do not guarantee better generalization capability. Despite learning well, the larger number of parameters in the deep model only resulted in easier overfitting of the training data and did not provide adequate generalization. While shallow CNNs do not suffer from overfitting as much as the deeper model, the Self-ONN variants, using the same number of trainable parameters, provide considerably better generalization performance.

5.4. Parameter–performance relationship curve

In Fig. 7, a plot of the number of trainable parameters and the corresponding generalization performance is shown for all networks used in this study for each of the three noise types. The Self-ONN networks, powered by the proposed generative neurons, consistently provide a better tradeoff between the number of parameters and the denoising performance as compared to the convolutional networks. Each of the Self-ONN variants outperforms the corresponding CNN with an equal number of parameters. This is a testament to the superior learning capability of the generative neurons. Moreover, as discussed earlier in Section 5.3, the 17-layer deep CNN (DnCNN) fairs the worst across all three noise types, despite having at least 19 times more trainable parameters than the competing networks. This can be attributed to the fact that enhanced generalization capacity cannot be realized by deeper models in the absence of more training resources.

5.5. Computational complexity analysis

5.5.1. Theoretical operations

Table 4 provides a comparison of the total number of multiply-accumulate (MAC) operations for the networks considered in this study. One MAC corresponds to a single multiply and addition operation, making it roughly equivalent to 2 FLOPS. The number of MACs for the lth layer of the network is calculated using the following formula:

$$MACs\ (l) = |Y_l| * \left((N_{l-1} * K_x^l * K_y^l * Q^l) + 1 \right)$$

where $|Y_l|$ is the number of elements in the output of the current layer, N_{l-1} is the number of neurons in the previous layer, K_x^l and K_y^l are the kernel dimensions for the current layer, Q^l is the order of approximation. The last term can be omitted for the special case where the bias is not used. Besides, for each network, we also calculate the ratio of MACs and the average generalization performance in terms of PSNR. This provides a quantification of the utility of trainable parameters in each network architecture.

As expected, we see that Self-ONNs have approximately the same number of MACs as compared to their equivalent CNNs. However, the highest order Self-ONN, Self-ONN-7, requires 4.65G MACs per dB of test PSNR, whereas the next lower order configuration, Self-ONN-5, only consumes 1.99G flops per dB of generalization performance. Two key insights can be gained from this observation. Firstly, the difference implies that there exists a certain optimal value of Q for a given problem for Self-ONNs, above which the generalization performance saturates for this particular problem. Secondly, there might exist a sparsity in the higher-order weights, which can be exploited during inference to

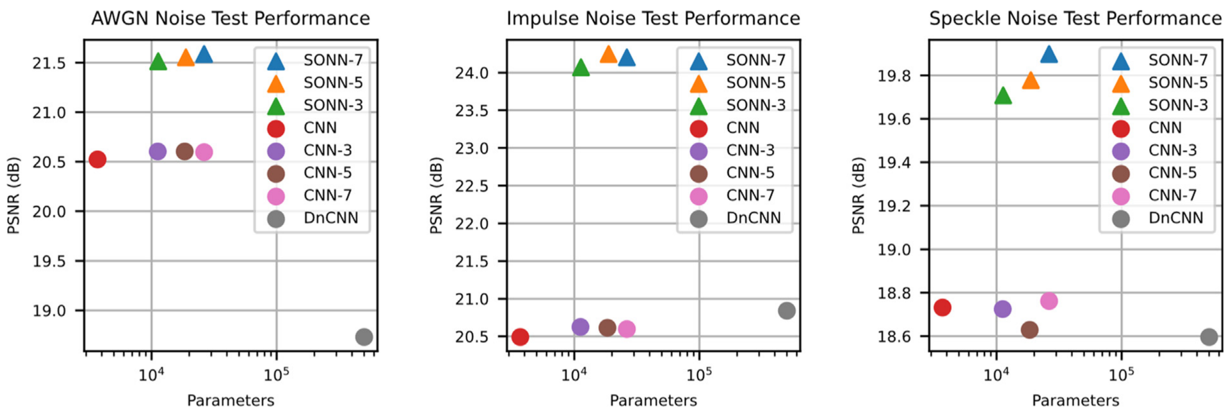

Fig. 7. Test set performance (averaged across all datasets) versus the number of parameters of the networks used in this study. Networks with generative neurons (Self-ONNs) are represented by triangles while the ones with convolutional neurons (CNNs) are represented by circles.

Table 4
The computational complexity of each network in terms of multiply-accumulate (MAC) operations.

	MACs (G)	MACs per dB (G/dB)
Self-ONN-7	94.71	4.655
Self-ONN-5	67.66	1.998
Self-ONN-3	40.62	3.27
CNN-3	40.41	4.961
CNN-5	66.28	2.138
CNN-7	94.61	3.562

Table 5
Running time comparison.

Network	256 × 256	512 × 512	1024 × 1024
Self-ONN-7	0.13	0.48	1.93
Self-ONN-5	0.09	0.36	1.43
Self-ONN-3	0.05	0.22	0.88
CNN-7	0.06	0.21	0.84
CNN-5	0.04	0.18	0.71
CNN-3	0.03	0.15	0.58
ONN	0.91	3.81	10.11
DnCNN	0.41	1.65	6.70

speed up performance using only a subset of the Q-dimensional weights, at the expense of negligible loss of performance. Both of these observations will be investigated further in our future studies.

5.5.2. Execution time

Table 5 presents the inference time comparison for all the models used in this study with different image sizes. The benchmarking was performed on an NVIDIA GTX 1060 GPU with CUDA version 10.1 running on a 7th-generation i7 Intel processor. The ONN and Self-ONN networks were implemented using FastONN (Malik, Kiranyaz, & Gabbouj, 2020) library which utilizes PyTorch (Paszke et al., 2019).

From Table 5, we notice that the execution time for the Self-ONN is considerably lower than the one for the vanilla ONNs. As discussed in Section 3.3, this is because of the fact that Self-ONNs can be expressed in terms of convolutions, which lends them to faster computation on GPUs. Moreover, on average, a Self-ONN with equivalent parameters is 0.04, 0.17, and 0.7 s slower for images of sizes 256 × 256,512 × 512 and 1024 × 1024 respectively, as compared to a CNN. While such differences are practically negligible if the gain in generalization performance is considered, it is nevertheless worth noting that the Self-ONN GPU implementation is not yet optimized. Specifically, the Hadamard exponentiation operation of (8) and the subsequent convolution operations are carried out by different CUDA kernels. This adds the overhead of launching kernels as-well-as copying data to-and-from the GPU. Fusing the two kernels would enable efficient exploitation of the fact that the number of independent operations in a convolutional neuron and generative neurons will be the same, and would result in a significantly reduced inference time. This will be a part of our focus for future extensions.

6. Conclusions

In this study, we propose Self-ONNs to tackle severe image restoration problems. Self-ONNs are composed of generative neurons, which have the ability to synthesize any nodal operator by leveraging Taylor polynomials. Our results provide conclusive evidence that these optimized nodal transformations achieved, through generative neurons, considerably higher learning and generalization performance when compared with the linear mappings of convolutional neurons. Moreover, we show that adding learnable parameters by increasing the number of convolutional filters is sub-optimal while doing so in a manner such that the added parameters can more directly influence the degree of non-linearity is a more worthy investment. The insufficient enrichment of the solution space when adding convolutional filters is one of the key reasons why state-of-the-art CNN architectures for image restoration are generally very deep. Finally, we show that the proposed Self-ONNs can even outperform its predecessor, ONNs, suggesting that for challenging inverse imaging problems such as the ones tackled in this study, hand-crafting an operator set library is not practical as the required nodal transformation may not exist in the form of well-known functions.

The core idea of generative neurons provides a modular interface and as such, it can be incorporated directly into the prevalent denoising and restoration architectures to increase their performance, as well as decrease the network size. Moreover, the networks used in this study were compact and used basic training paradigms such as SGD-based optimization. It will be an interesting research direction to explore various contemporary techniques for Self-ONNs essentially used for stabilizing the training of CNNs, such as batch normalization, Adam optimization with its variants, and dropout. These will be the subjects of our future research direction.

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.

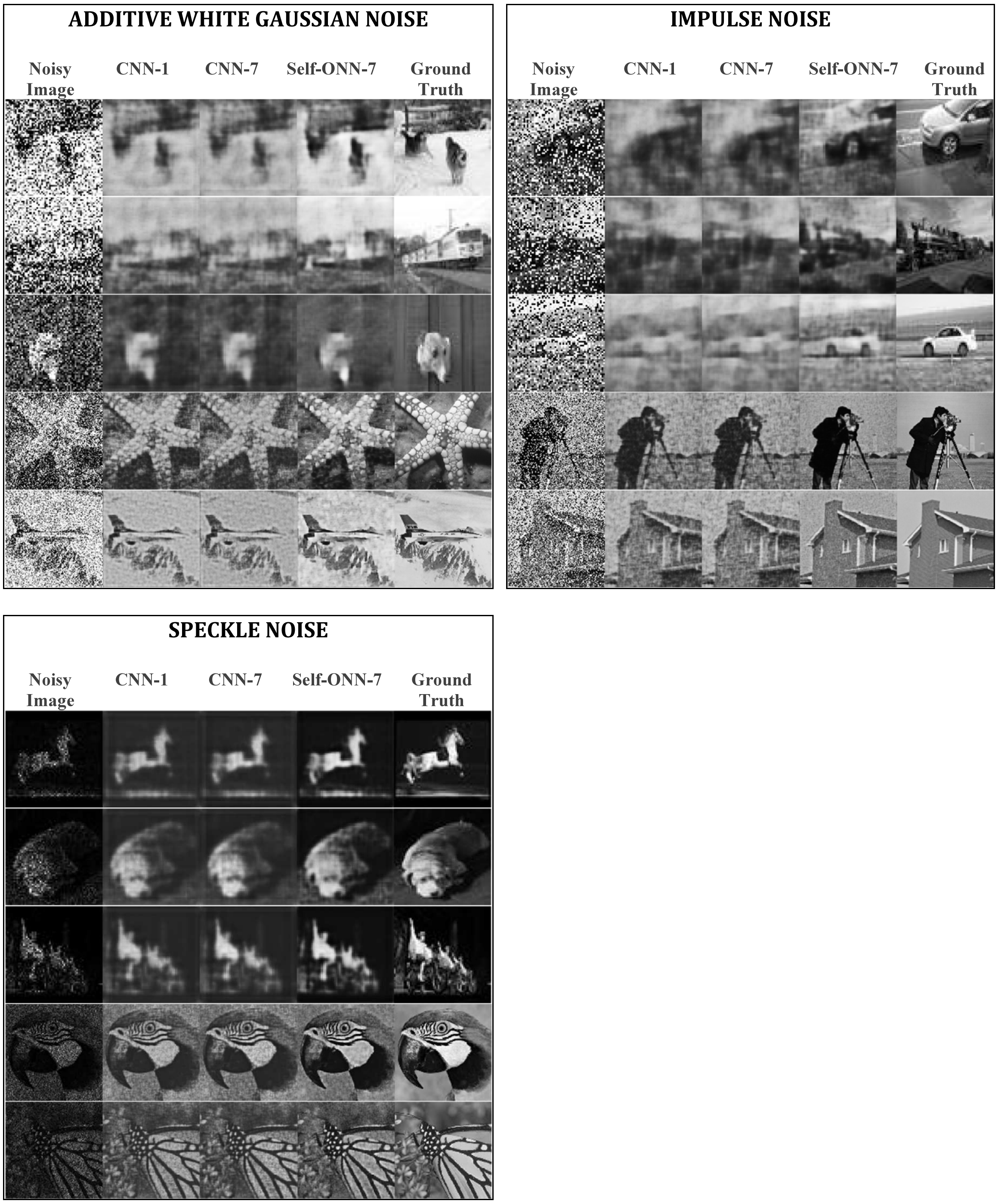

Fig. 8. Sample restoration results from Pascal (patches) and BSD12 (high-resolution) datasets comparing the restoration performances.

Acknowledgments

This work was supported by the Qatar National Research Fund (QNRF) through the ongoing project under Grant NPRP11S-0108-180228. Funding from Academy of Finland AWcHA and Business Finland Amalia is acknowledged.

References

Alvarez, L., Lions, P. L., & Morel, J. M. (1992). Image selective smoothing and edge detection by nonlinear diffusion. II. *SIAM Journal on Numerical Analysis*.

Bioucas-Dias, J. M., & Figueiredo, M. A. T. (2010). Multiplicative noise removal using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*.

Buades, A., Coll, B., & Morel, J. M. (2005). A non-local algorithm for image denoising. In *Proceedings - 2005 IEEE computer society conference on computer vision and pattern recognition*.

Burger, H. C., Schuler, C. J., & Harmeling, S. (2012). Image denoising: Can plain neural networks compete with BM3D? In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Chetlur, S., et al. (2014). cuDNN: efficient primitives for deep learning.

Cires, D. C., Meier, U., Masci, J., & Gambardella, L. M. (2003). Flexible, high performance convolutional neural networks for image classification. In *Proceedings of the twenty-second international joint conference on artificial intelligence*.

Conneau, A., Schwenk, H., Le Cun, Y., & Barrault, L. (2017). Very deep convolutional networks for text classification. In *15th Conference of the European chapter of the Association for computational linguistics, EACL 2017 - Proceedings of conference*.

Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*.

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transaction on Information Theory*.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic gradient descent. In *ICLR int. conf. learn. represent*.

Kiranyaz, S., Ince, T., Iosifidis, A., & Gabbouj, M. (2017a). Generalized model of biological neural networks: Progressive operational perceptrons. In *Proceedings of the international joint conference on neural networks*.

Kiranyaz, S., Ince, T., Iosifidis, A., & Gabbouj, M. (2017b). *Generalized model of biological neural networks: Progressive operational perceptrons, vol. 1* (pp. 2477–2485).

Kiranyaz, S., Ince, T., Iosifidis, A., & Gabbouj, M. (2020). Operational neural networks. *Neural Computing and Applications*.

Kiranyaz, S., Malik, J., Abdallah, H. B., Ince, T., Iosifidis, A., & Gabbouj, M. (2020). Exploiting heterogeneity in operational neural networks by synaptic plasticity. *Neural Computing and Applications*, (November).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Alex Net. *Advances in Neural Information Processing Systems*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*.

Lempitsky, V., Vedaldi, A., & Ulyanov, D. (2018). Deep image prior. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Mahmoudi, M., & Sapiro, G. (2005). Fast image and video denoising via nonlocal means of similar neighborhoods. *IEEE Signal Processing Letters*.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2009). Non-local sparse models for image restoration. In *Proceedings of the IEEE international conference on computer vision*.

Malik, J., Kiranyaz, S., & Gabbouj, M. (2020). FastONN – Python based open-source GPU implementation for Operational Neural Networks. arXiv.

Paszke, A., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 8026–8037). Curran Associates, Inc..

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Smith, S. M., & Brady, J. M. (1997). SUSAN - A new approach to low level image processing. *International Journal of Computer Vision*.

Tian, C., Xu, Y., & Zuo, W. (2020). Image denoising using deep CNN with batch renormalization. *Neural Networks*.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5 - rmsprop. *COURSERA Neural Networks for Machine Learning*.

Tran, D. T., Kiranyaz, S., Gabbouj, M., & Iosifidis, A. (2020a). Heterogeneous multilayer generalized operational perceptron. *IEEE Transactions on Neural Networks and Learning Systems*.

Tran, D. T., Kiranyaz, S., Gabbouj, M., & Iosifidis, A. (2020b). Progressive operational perceptrons with memory. *Neurocomputing*.

Wang, C., Yang, J., Xie, L., & Yuan, J. (2019). Kervolutional neural networks. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Wu, X. (2011). Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic Imaging*.

Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*.

Zhang, K., Zuo, W., Gu, S., & Zhang, L. (2017). Learning deep CNN denoiser prior for image restoration. In *Proceedings - 30th IEEE conference on computer vision and pattern recognition*.

Zhang, K., Zuo, W., & Zhang, L. (2018). Ffdnet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*.

Zou, H., Lan, R., Zhong, Y., Liu, Z., & Luo, X. (2019). EDCNN: A novel network for image denoising. In *Proceedings - international conference on image processing*.

Zoumpourlis, G., Doumanoglou, A., Vretos, N., & Daras, P. (2017). Non-linear convolution filters for CNN-based learning. In *Proceedings of the IEEE international conference on computer vision*.