Article

Deep Self-Organizing Map of Convolutional Layers for Clustering and Visualizing Image Data

Christos Ferles *, Yannis Papanikolaou, Stylianos P. Savaidis and Stelios A. Mitilineos

Department of Electrical and Electronics Engineering, University of West Attica, GR-12241 Aegaleo, Attica, Greece; ypapanikolaou@uniwa.gr (Y.P.); ssavaid@uniwa.gr (S.P.S.); smitil@uniwa.gr (S.A.M.)
* Correspondence: christos.ferles@gmail.com

Abstract: The self-organizing convolutional map (SOCOM) hybridizes convolutional neural networks, self-organizing maps, and gradient backpropagation optimization into a novel integrated unsupervised deep learning model. SOCOM structurally combines, architecturally stacks, and algorithmically fuses its deep/unsupervised learning components. The higher-level representations produced by its underlying convolutional deep architecture are embedded in its topologically ordered neural map output. The ensuing unsupervised clustering and visualization operations reflect the model's degree of synergy between its building blocks and synopsize its range of applications. Clustering results are reported on the STL-10 benchmark dataset coupled with the devised neural map visualizations. The series of conducted experiments utilize a deep VGG-based SOCOM model.

Keywords: deep learning; unsupervised learning; convolutional neural network (CNN); self-organizing map (SOM); clustering; visualization

check for updates

Citation: Ferles, C.; Papanikolaou, Y.; Savaidis, S.P.; Mitilineos, S.A. Deep Self-Organizing Map of Convolutional Layers for Clustering and Visualizing Image Data. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 879–900. https://doi.org/10.3390/make3040044

Academic Editor: Randy Goebel

Received: 21 October 2021
Accepted: 12 November 2021
Published: 14 November 2021

1. Introduction

For more than a decade, deep learning has been at the forefront of the development of methods that shift the focus towards meaningful representation discovery by algorithms. The devised distributed layered representations, which build upon lower-level invariant partial features, reveal higher-level abstract concepts and aspects of the data. The induced responses, from discovered correlations within data, depend on the connectivity and memory characteristics of the neurons. In an algorithm this is implemented as multiple sequential causative compute events wherein each event transforms (often in a nonlinear way) the aggregate response of the network [1]. Deep learning within this context refers to the accurate adjustment of parameters (weights and biases) across such events.

Probably the most common bottleneck encountered in many deep learning approaches like convolutional neural networks (CNNs) is the requirement for big labeled datasets. Constructing these datasets is a time-consuming costly procedure that frequently might end up proving error-prone or even infeasible for various reasons. Even commonly used computer vision datasets have been shown to be susceptible to such label errors [2]. The obvious (but not necessarily simple) answer to these problems is devising deep learning models that can be trained with unlabeled/uncategorized data; in other words, invent unsupervised learning algorithms for such deep networks. Aligned with this ongoing research direction one can trace a number of works that combine or hybridize self-organizing maps (SOMs) with CNNs. The common denominator in these models is to equip CNNs with the unsupervised clustering capabilities of the SOMs, or inversely, extract deep representations (e.g., CNN codes) and quantize them into the SOM neural map.

The range of these approaches—including the present one—is quite widespread, spanning the range from purely unsupervised learning algorithms up to semi (or even full) supervised ones, and from shallow networks and growing/hierarchical models [3–6] to architectures containing multiple hidden layers; for instance [7–11]. Meeting both main objectives i.e., building a deep SOM and training it in a purely unsupervised way has proven a complex and difficult task. Only a small number of models exist that can be classified as unsupervised beyond any doubt [12–14]. Equally few are the approaches that extend beyond the three hidden layer limit [13,15,16].

In a nutshell, the key characteristics and contributions of the proposed self-organizing convolutional map (SOCOM) prototype are:

(1) A generic deep convolutional architecture that extends far beyond the trivial three hidden layer limit of shallow networks.
(2) An inherent flexibility to embed existing deep convolutional models and to facilitate transfer learning from pre-trained CNNs, these can be used either as fixed feature extractors (yielding CNN codes) or as initial weight/parameter values for the subsequent backpropagation stages.
(3) An end-to-end unsupervised learning algorithm that does not necessitate the targets/labels of the training samples at any stage, and is specifically tailored to meet the requirements of the architecture's complexity, depth, and parameter size.
(4) A complementary neural map visualization technique that offers insight and interpretation of the SOCOM clusters, or equivalently, a projection and quantization of the achieved higher-level representations onto the array of output neurons; this is also achieved without using any type of label information throughout the respective processes.

The organization and structure of the remaining four sections of this paper are as follows. Section 2 presents in detail and in-depth the SOCOM both architecturally and algorithmically. Subsequently, this section analyzes the key components of the learning and feedforward operations. Section 3 contains experimental results with the focus on the analysis and systematic evaluation of the devised neural map visualization technique. In addition, experimental comparisons are carried out against several related algorithms on a deep learning benchmark dataset. Section 4 summarizes the whole paper and draws conclusions.

2. SOCOM Prototype

A generic and at the same time characteristic SOCOM architecture consisting of multiple convolutional, pooling, fully connected, and self-organizing layers is illustrated in Figure 1. The mathematically expressed algorithmic learning procedures are presented in the following subsection. This section describes the main functionality and key methods of the SOCOM from a macroscopic operational point of view.

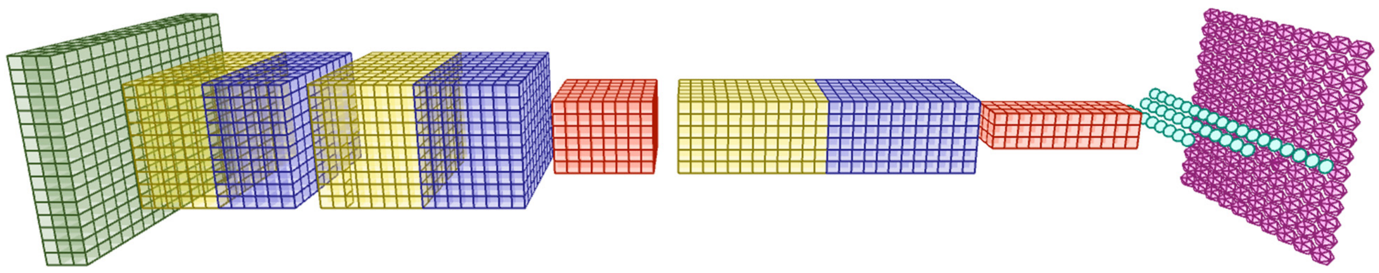

Figure 1. Detailed architecture of a SOCOM paradigm consisting of an input layer (green), 3 convolutional layers (yellow) followed by ReLUs (blue), 2 pooling layers (red), 3 fully connected layers (turquoise), and an output neural map (purple).

The input layer of the SOCOM accepts any type of numerical data arranged in vectors, matrices (e.g., grayscale images), or volumes (e.g., colored images or successive images that exhibit a spatiotemporal correlation). The explicit assumption of CNNs that the inputs' elements are correlated, something that makes the information propagation more efficient and hugely reduces the network's parameter count, still holds in the SOCOM paradigm but does not a priori exclude all other types of input data.

As can be seen, a SOCOM comprises a sequence of different layers with adjustable parameters. Each respective layer transforms one volume of activations to another via a differentiable function, thus facilitating the use of backpropagation during training. Stacking these layers in series eventually forms a full SOCOM architecture (Figure 1).

A SOM lattice of topologically arranged neurons acts as the output layer. Each of its neurons receives the activations of every unit in the last fully connected layer. The magnitude of each neuron's activation is based on a distance metric between the input activations and its codebook parameters. The neural mapping of the input image coincides with the position of the neuron that produces the optimal fit with respect to the computed activations and the neighborhood kernel (which has been defined over the topology of the neural grid). Apart from mapping, this particular type of nonlinear projection can be further exploited for data clustering and visualization.

It is also interesting to note that the proposed SOCOM architecture is in a position to incorporate any number of layers (from the previous types) in any permutation. There are only two limitations: (1) after the first fully connected layer convolutional layers cannot be used, (2) the output layer needs to be a SOM grid.

With respect to Figure 1, let a sample be applied to the inputs of the SOCOM. A kernel which is a part of the first (hidden) convolutional layer computes its activations by sliding its receptive field along the width and the height of the input volume and by applying a nonlinearity. This process is repeated for all the filters that form the first convolutional layer. After all the activations have been gathered, they are arranged in a feature map which is considered to be the input volume for the following layer. If the next layer is a convolutional layer the previous process is repeated. If it is a pooling layer then the input volume is downsampled along its width and height spatial dimensions but not along its depth. When the representations of the last convolutional (or pooling) layer have been computed then all the activations of the corresponding feature map are connected to every unit in the fully connected layer. Its units perform affine transformations and their activations are calculated by applying a nonlinear squashing function to the results of the transformations. Once again, this procedure is repeated per layer until the defined number of fully connected layers has been incorporated. In the end, the neurons of the output lattice receive the activations from the last hidden fully connected layer. By taking into consideration their codebook weights and their position onto the grid an activation (or response) is computed. After comparing all the activations, the neuron (viz. its position onto the self-organizing grid) yielding the optimal response identifies with the projection of the input sample onto the output layer. Clusters around paradigms (encoding underlying patterns and distributions) of the input samples are formed by accumulating their respective projections onto the output plane.

SOCOM transforms the initial input image layer by layer to the final output mapping. The layers that contain tunable parameters are the convolutional, the fully connected, and the self-organizing; the gradient descent backpropagation algorithm is utilized for performing the necessary learning adjustments. On the contrary, the ReLU and pooling layers do not require any training because they implement fixed functions that do not have any modifiable parameters.

2.1. SOM Review

Studies have convincingly shown that the best self-organizing results are obtained if the following two partial processes are implemented in their purest forms [17]:

(1) Decoding of that neuron that has the best match with the input data pattern (the so-called winner);
(2) Adaptive improvement of the match in the neighborhood of neurons centered around the winner.

The SOM may be described formally as a nonlinear, ordered, smooth mapping of input data onto the elements (denoted as e) of a regular, low-dimensional array. The mapping is implemented in the following way, which resembles the two aforementioned processes. Assume first that x is an input vector. With each element e in the SOM array a vector u_e (codebook) is associated. Considering the Euclidean distances of x given each u_e the image of the input vector on the SOM array is defined as the neuron (denoted as c) yielding the smallest Euclidean distance:

$$c = \underset{e}{argmin} \left|\left| x - u_e \right|\right|. \tag{1}$$

Subsequently, the classical rule for updating the neurons' codebook parameters is:

$$u_e^{(next)} = u_e + \eta h_{c,e}(x - u_e) \tag{2}$$

where η is the learning rate and $h_{c,e}$ is the neighborhood function/kernel. The core idea is to optimize proportionally the parameters of the neurons lying in the vicinity of the winner so as to gain some knowledge from the same input x.

The SOM, in its basic form, produces a nonlinear projection of input data. It converts the complex statistical relationships between data into simple geometric relationships of their image points on a low-dimensional display, usually a regular two-dimensional grid of neurons. As the SOM thereby compresses information, while preserving the most important topological and statistical relationships of the primary data elements on the display, it may also be thought to produce some kind of abstractions. These characteristics, abstraction, dimensionality reduction, and visualization in synergy with clustering, have been utilized in a widespread and extensive set of data analysis tasks.

2.2. *Forward Propagation*

A generic SOCOM architecture consists of an input layer, L hidden layers (convolutional, pooling, and fully connected ones), and an output layer (viz. lattice of ordered neurons).

2.2.1. Convolutional Layer

$w^{l,p}_{m,n,d}$ is the m-th, n-th, and d-th element of weight matrix of filter p connecting neurons of layer l with neurons of layer $l-1$. Kernel or filter p is of dimension $k_1 \times k_2 \times D$. Consequently, at each layer l for the bank of P filters we have $w \in \mathbb{R}^{k_1 \times k_2 \times D \times P}$ and biases $b \in \mathbb{R}^P$. At such a layer a convolution operation is carried out (e.g., Figure 2), which is the same as a cross-correlation with a rotated kernel. The convolved input vector of filter p at layer l plus the bias is represented as $x^{l,p}_{i,j}$ and is calculated according to:

$$x^{l,p}_{i,j} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{d=0}^{D-1} w^{l,p}_{m,n,d} O^{l-1,d}_{i+m,j+n} + b^{l,p} \tag{3}$$

where $O^{l-1,d}$ is the output of the d-th filter at layer $l-1$. Particularly at the first layer (viz. the input layer) we feed an image (or a sequence of images) with height H_1, width H_2 and depth D such that $I \in \mathbb{R}^{H_1 \times H_2 \times D}$. At the first hidden convolutional layer this results in:

$$x^{1,p}_{i,j} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{d=0}^{D-1} w^{1,p}_{m,n,d} I_{i+m,j+n,d} + b^{1,p}. \tag{4}$$

Frequently the convolution layer is coupled with (viz. followed by) a non-saturating activation function which is applied element-wise thresholding to zero (Figure 2). The ReLU activation function induces sparsity to the hidden units thus resulting in more valuable representations. The output at layer l is the outcome of the application of the activation layer to the convolved layer:

$$O_{i,j}^{l,p} = f\left(x_{i,j}^{l,p}\right) = max\left(0, x_{i,j}^{l,p}\right) = \begin{cases} x_{i,j}^{l,p}, & x_{i,j}^{l,p} > 0 \\ 0, & x_{i,j}^{l,p} \leq 0. \end{cases} \tag{5}$$

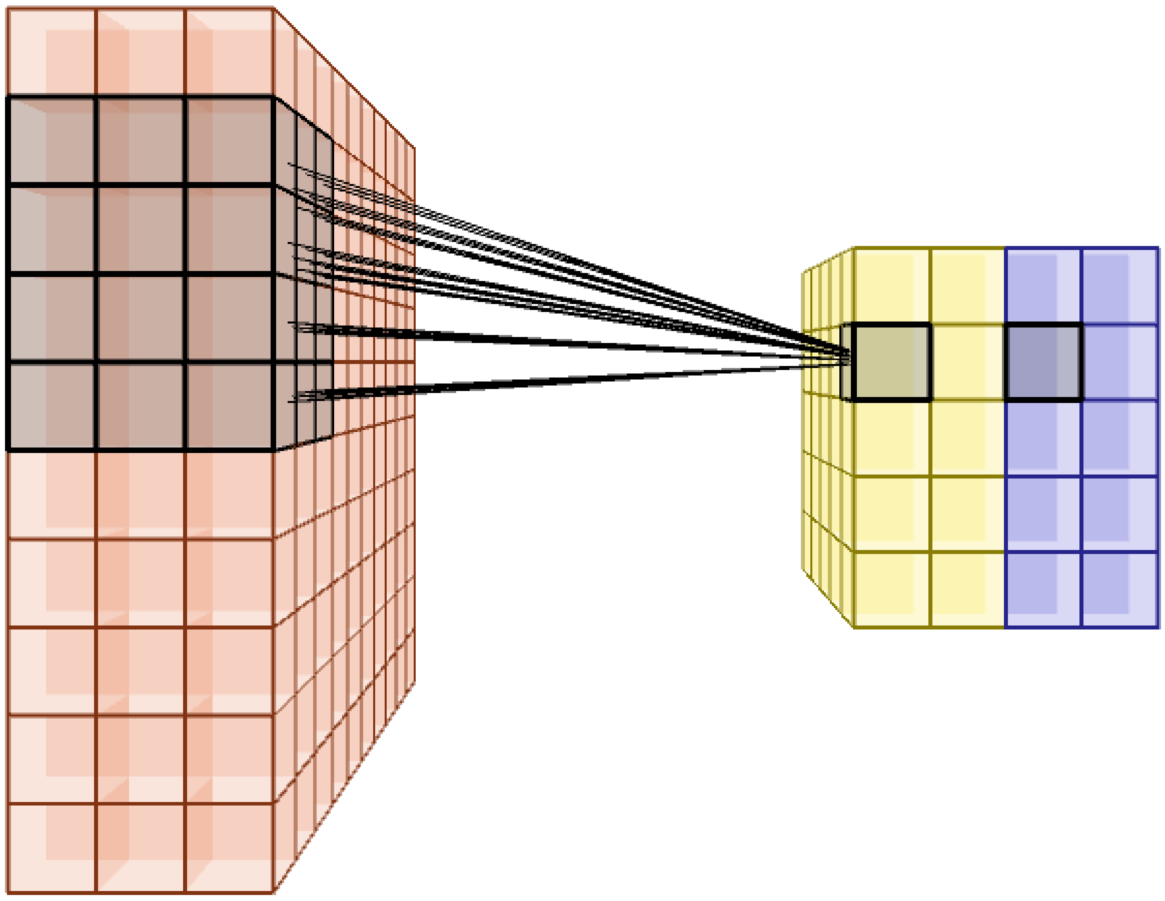

Figure 2. Example of a convolutional layer comprised of 2 filters (yellow) that are applied to a 3 channel input volume. Subsequently, each element of the resulting feature maps is fed through a ReLU (blue).

2.2.2. Pooling Layer

Periodically a pooling layer is inserted in between successive convolutional layers (Figure 3), its aim is to progressively reduce the spatial size of the representation; thus, (a) reducing the number of parameters and computation in the following layers and (b) controlling overfitting. The max operation is used more frequently, other types of pooling like average or L_2 norm pooling have been shown to not work equally well. Let a pooling layer of size $k_p \times k_p$ that slides over its input with a stride equal to s_p thus reducing $k_p \times k_p$ blocks to a single value.

The outcome of the pooling layer is calculated according to:

$$O_{i,j}^{l,p} = \max_{0 \leq a \leq k_p - 1, 0 \leq b \leq k_p - 1} \left(O_{i \cdot s_p + a, j \cdot s_p + b}^{l-1,p}\right). \tag{6}$$

Nearly always the choice at the pooling layer is either a 2×2 region filter with a stride of 2 or an overlapping pooling operation with a 3×3 filter size and a stride of 2.

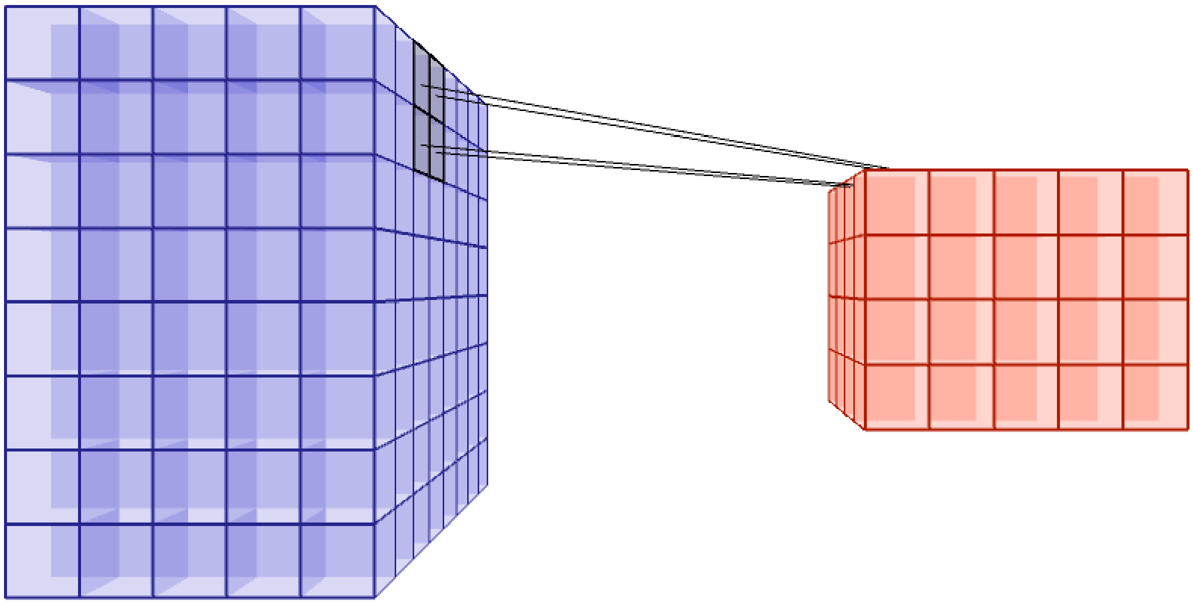

Figure 3. Example of a pooling layer (red) connected after a convolutional or ReLU layer comprised of 5 feature maps.

2.2.3. Fully Connected Layer

In this case, each unit at a given layer l is connected to every unit in the previous layer $l-1$. The weight (or parameter) associated with the connection between unit $j's$ output (at layer $l-1$) and the unit i in layer l is denoted as $w^l_{i,j}$. Additionally, b^l_i is the bias associated with unit i in layer l. Apart from the ReLU other common choices for the nonlinear activation function $f()$ (particularly in multilayer perceptrons and autoencoders) are the sigmoid and the hyperbolic tangent. The computation that each individual unit represents is essentially a weighted sum of the unit's inputs including the bias term:

$$x^l_i = \sum_{j=0}^{P-1} w^l_{i,j} O^{l-1}_j + b^l_i \tag{7}$$

$$O^l_i = f\left(x^l_i\right) \tag{8}$$

where P is the total number of units in layer $l-1$. As can be seen, starting with some set of activations from the previous layer the inputs to the units at the next layer are computed (Figure 4) and after applying the nonlinearity this pattern of propagation is continued until the desired layer is reached.

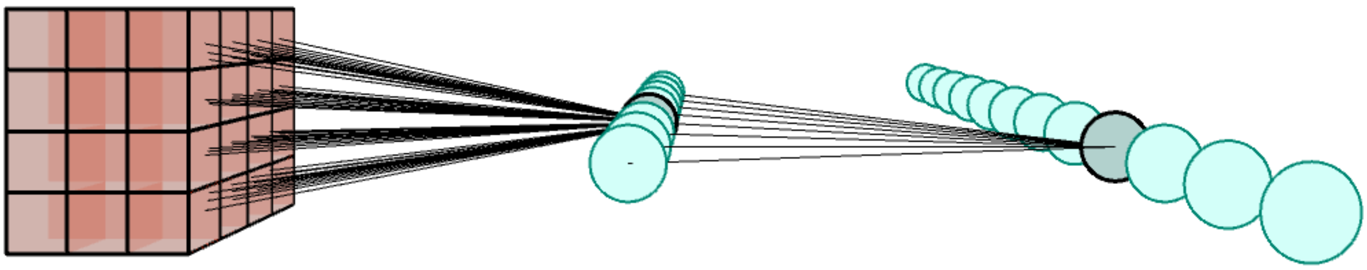

Figure 4. Example of 2 successive fully connected layers (turquoise) following a convolutional-ReLU or pooling layer (red).

Between the last convolutional layer (or probably, the last pooling layer) and the first fully connected layer, a different connectivity pattern exists between the elements and the units of the underlying and the overlying layers (Figure 4). Practically, in terms of formulation this can either be accomplished by algorithmically converting fully connected layers to convolutional layers or alternatively by squashing the feature maps' elements into a single vector:

$$O^{l}_{j+i\cdot H_2+p\cdot H_1\cdot H_2} = O^{l,p}_{i,j} \tag{9}$$

where, in this particular case, l is assumed to be the last convolutional layer consisting of $H_1 \times H_2$ feature maps.

2.2.4. Output Layer

The output layer that consists of G topologically arranged neurons performs a mapping of its input representations onto its neural map (Figure 5). More specifically, the projection of an input representation on the SOCOM plane is defined as the neuron yielding the lowest weighted squared Euclidean distance between the last hidden layer's outputs O^L_i and its corresponding codebook parameters $u_{g,i}$ where weighting refers to the neighborhood kernel/function $h_{e,g}$ defined over the topology of the neural grid. Frequently, this neuron (denoted as c) is referred to as the winner. Algorithmically, this best-matching winner neuron is given by:

$$c = \underset{e}{argmin} \sum_{g=0}^{G-1} h_{e,g} \sum_{i=0}^{P-1} \left(O^L_i - u_{g,i}\right)^2 \tag{10}$$

where P is the total number of units in the last layer L. Additionally, this particular type of nonlinear projection can be further exploited for data clustering and visualization procedures. Additionally, if the unimodal neighborhood function's radius is narrow enough, so as to contain mainly the closest neighbors, then in the landslide of cases the previously detected best-matching neuron coincides with the usual winner neuron of the original SOM learning algorithm.

More particularly, in the mapping and in the (complementary) learning processes the function $h_{e,g}(y)$ has a very central role; it acts as the neighborhood function, a smoothing kernel defined over the lattice neurons. y symbolizes time, or equivalently, the corresponding iteration. For convenience, it is necessary that $h_{e,g}(y) \to 0$ when $y \to \infty$. Usually $h_{e,g}(y) = h(||r_e - r_g||, y)$, where $r_e, r_g \in \mathbb{R}^2$ are the location vectors of neurons e and g on the lattice. With increasing $||r_e - r_g||$ the function $h_{e,g}(y) \to 0$. The width and form of $h_{e,g}(y)$ define the stiffness of the elastic surface to be fitted to the input representations. In the literature, the most frequently used neighborhood kernel can be written in terms of the Gaussian function:

$$h_{e,g}(y) = exp\left(-\frac{||r_e - {r_g}^2||}{2\sigma(y)^2}\right) \tag{11}$$

where the square root of the variance $\sigma(y)$ defines the width of the kernel (radius) and is a monotonically decreasing function of time.

2.3. Backpropagation

The purpose of being in a position to compute the error is dual. First, a quantification/estimation of the network's performance is obtained. Second, learning takes place via the optimization of the network's weights to minimize this specific error. This error function can be a number of different things, such as binary cross-entropy or sum of squared residuals. Differently from supervised approaches, learning in the case of SOCOM does not necessitate any type of desired or target values at any stage; thus giving rise to an end-to-end unsupervised deep learning algorithm. The corresponding error/cost/loss function (or alternatively, the penalty term) is symbolized as E and is defined as:

$$E = \sum_{c=0}^{G-1} N(c) \sum_{d=0}^{G-1} h_{c,d} \frac{1}{2} \sum_{i=0}^{P-1} \left(O_i^L - u_{d,i}\right)^2 \tag{12}$$

where

$$N(c) = \begin{cases} 1, & c = \underset{e}{argmin} \sum_{d=0}^{G-1} h_{e,d} \frac{1}{2} \sum_{i=0}^{P-1} \left(O_i^L - u_{d,i}\right)^2 \\ 0, & otherwise. \end{cases} \tag{13}$$

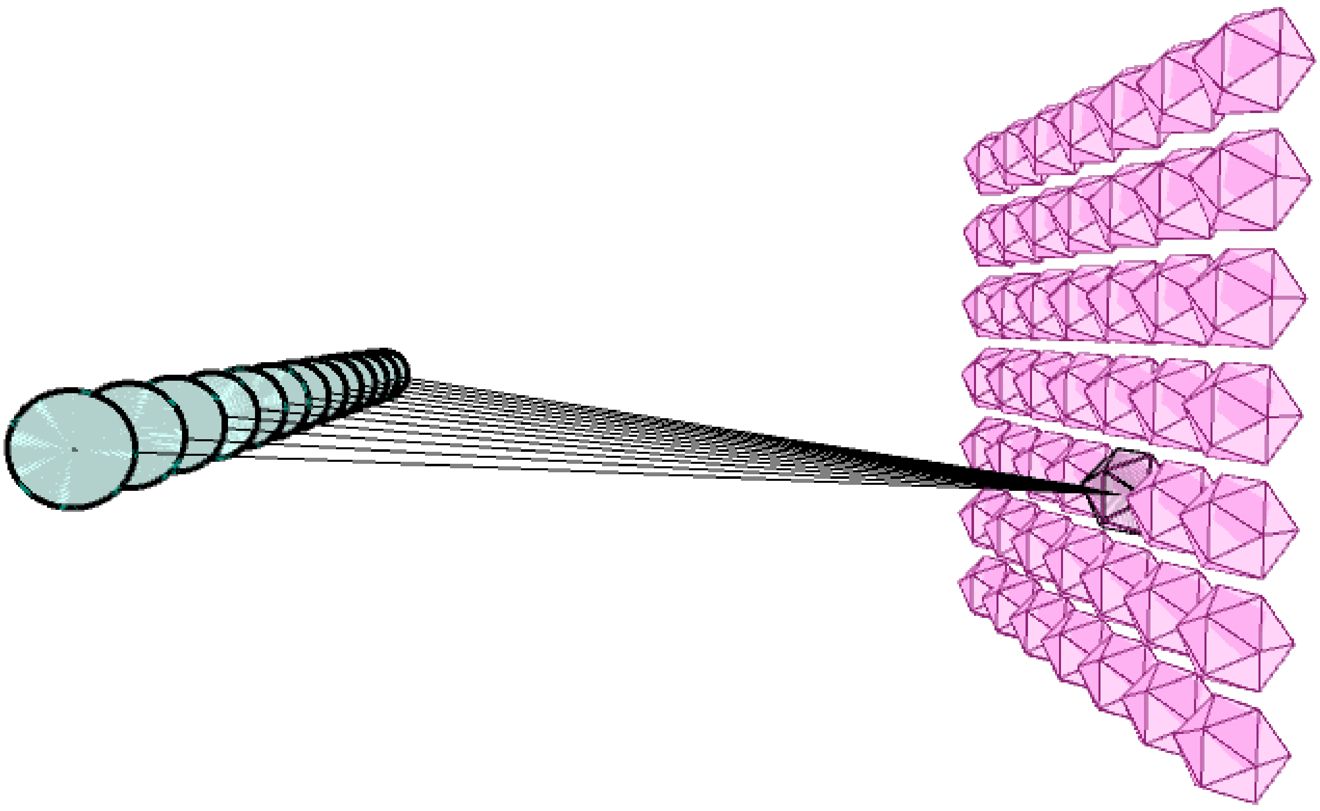
Figure 5. Example of an output layer consisting of neurons arranged onto an orthogonal lattice (purple). For performing the designed projection, each individual neuron receives the activations from units of the last fully connected layer (turquoise).

For gradient descent backpropagation the updates that need to be performed are for the weights, the biases, and the deltas (i.e., the tunable parameters of the SOCOM algorithm). The utilized energy formula by the SOCOM is in accordance with the variation proposed in [18] and has been also adopted by our previous hybrid SOM networks [19,20].

The benefits of the utilized energy function are noticeable: (1) The derived learning equations are no longer heuristic (as in the classical SOM approaches) but instead they are fully proven mathematically. (2) By conceptualizing (a priori) what in fact the training rules minimize, one has access to a global measure of learning performance. (3) Since due to its construction the cost function is differentiable, the corresponding partial, and total derivatives can be computed in a straightforward way, something that provides the capability to devise gradient backpropagation-based training algorithms.

In general, the benefit of having a differentiable loss function for a model currently becomes even more important since the two major machine learning libraries Pytorch and Tensorflow have built-in capabilities for automatic differentiation (torch.autograd and tf.GradientTape, respectively). For instance, according to Pytorch's documentation "torch.autograd provides classes and functions implementing automatic differentiation of arbitrary scalar valued functions". The automatic differentiations of all operations on tensors simplify the required backward executions/passes. This facilitates the realization of gradient backpropagations which are essential parts for a number of stochastic gradient descent learning/optimization algorithms. As a result, the synergy of the SOCOM's loss function with the automatic differentiation capabilities of Pytorch brings forth optimization/learning capabilities that were not applicable to SOM approaches of the past.

3. Experiments

3.1. Neural Output Visualization

Intrinsically, the spatial arrangement of the neurons in the output plane of the classical SOM lends itself to a wide and rather diverse range of techniques that aim to visually present aspects of the trained model's projections and clustering results. The two-dimensional neural planes (and less frequently the three-dimensional neural volumes) of the SOM outputs that differentiate them from other well-established clustering algorithms, provide the basis for analyzing/summarizing domain space information, interpreting the produced results, studying the hidden relations, and drawing conclusions on the underlying (possibly latent) structures and patterns of the data under consideration. As a result, during the past years, there has been a constant flow of SOM-specific visualization techniques being published. U-matrix, P-matrix [21], U*-Matrix [22], cumulative/stacked representation planes [20], sequence likelihood projection [23], connectivity strength matrix visualization [24], Clusot surfaces [25], gradient fields and borderline visualizations [26], visualization induced SOMs [27], smoothed data histograms [28], and component planes and response surfaces [29], are only a few of the techniques that have been proposed during the past two decades. The common denominator in the above and similar approaches is that they exploit the structurally direct connections between the inputs and the outputs for devising projections onto the output grids, for enriching and refining the produced clusterings, and for demonstrating data relationships and patterns visually. Apart from those that solely operate on the output layer and fully ignore input information, like for instance the U-matrix, the rest are not applicable to SOMs with deeper architectures. The techniques based both on the neural output and the input feature space, that exploit their in between relationships and correlations, fail in the cases where the gradual architectural shifting from no hidden layers to deep networks makes the input space–output plane correspondences hard to detect and quantify.

On the contrary, there has been a number of successful approaches when it comes to understanding and gaining insight into what the various features and representation layers of CNNs encode. These techniques, exactly because CNN architectures are specifically tailored for images (or image-like input data), produce results in the form of images that are readily interpretable by humans; something which is usually not possible for other types of data.

The underlying idea in [30] is to find the input features' values (i.e., patterns) that maximize the activation of each specific neuron along the CNN architecture. Extending the activation maximization idea [31] described a technique for visualizing the class models (i.e., output layer) by computing an appropriately regularized image. The authors of [32] further refined this approach to incorporate the activations of each neuron to different types of features; its multiple facets were used to create a synthetic visualization. A number of additional regularization methods to bias images towards being more visually interpretable are contained in [33]. In addition to richer regularizers (viz. total variation, jitter) the work in [34] follows a per-layer response inversion approach (using natural pre-images) to gain insight into what a CNN models. A complementary technique is visualizing the interactions between neurons (viz. activation space) in an effort to better understand neural networks [35]. This is extended by visualizing groups of neurons that are together strongly activated [36]. The activation atlas [37] is the result of visualizing the space jointly represented by common interactions between neurons. On a slightly different path, a technique is proposed in [38] that utilizes a multilayered deconvolutional network to project the representation activations back to the input pixel space so as to trace the activity within the model in a visually interpretable way. The work in [39] proposes a visualization method that detects/highlights which pixels of an input image are particularly influential (or not at all influential) for a node in the network. The problem of estimating the contributions of a feature to the overall classification score is also examined in [40] and these contributions are further visualized as heatmaps.

The devised neural map visualization (NMV) uses the activation maximization technique as its main building block and aligns it with the structural and algorithmic characteristics of the neural output map. As a visualization mechanism, its goal is to provide insight into the inferred representations and clustering results of the SOCOM. With regards to the published methodologies, it follows the same substratal reasoning that a pattern to which a neuron responds maximally is a reasonable approximation of what a unit is doing. For each neuron g in the SOCOM output layer, the optimization problem posed it to find the image(s) $\ddot{y}$ that:

$$s = \left\{ j : arg \underset{0 \le j \le P}{topk} \left(u_{g,j}, k \right) \right\} \tag{14}$$

$$\ddot{y} = \underset{y}{argmax} \left(\sum_{i \in s} O_i^L(y) - \lambda ||y||_2^2 \right) \tag{15}$$

where P is the total number of units in the last hidden layer L, $u_{g,i}$ if the ith codebook parameter of neuron g, k represents the number of elements returned by the $topk()$ function that finds the top maximum valued elements in the given vector/matrix, and λ is the coefficient that controls the magnitude of the weight decay.

As can be seen, the optimization objective is comprised of a summation of the most important features fed to an output neuron coupled with an L_2 weight decay regulizer. The reasoning behind this strategy is that the SOCOM's output is based on Euclidean distances between codebook parameters and the last hidden layer's activations something that deviates from the mechanism found in the fully connected layers preceding the output; the highest valued codebook parameters of a specific neuron reveal which activations from the last hidden layer play a prominent role in rendering the specific neuron the best-matching winner, and consequently, they should be taken into consideration during the optimization process. It should be noted that such an approach is not entirely new since it is inspired by the supervised activation maximization counterparts where the (unnormalized) class scores are used instead of the class posteriors returned by the soft-max layer so as to avoid the phenomenon of maximizing the class posterior by minimizing the scores of other classes and not concentrating on maximizing the class in question.

A crucial problem that arises when activation maximization visualizations come into play is that "it is easy to produce images that are completely unrecognizable to humans, but state-of-the-art deep neural networks believe to be recognizable objects" [41]. Additional/alternative explanations of this phenomenon and of the closely related problem of adversarial examples' misclassification are given in [33,42,43]. The proven answer/solution as far as activation maximization is concerned is to impose regularizations during the optimization process to bias images in becoming more visually interpretable. When NMV was applied without a regularization method, the aforementioned problem also surfaced. In order to address it, certain types of regularization were brought into the test. Possibly the most popular in the literature i.e., the L_2 regularization, which tends to suppress the small number of extreme pixel values from distorting the output image, produced comparably better results. Furthermore, as can be seen in (15), the L_2 regularization was part of the objective function and was adjusted accordingly via the weight decay parameters of the SGD algorithm that was employed for performing the necessary optimization steps. On a side note, Adam and Adamax [44,45] both of which performed far better during the training procedures of SOCOM did not demonstrate equally higher performance and as a result, the simpler SGD [46] was qualified for the optimization required by the NMV.

The experimental setup and in particular the utilized dataset for the present series of experiments were chosen to serve a dual purpose. The first objective is to comply with the justifiable expectation of testing modern models on datasets that exhibit a certain level of difficulty and complexity. In particular, when it comes to images, the starting point during the proof-of-concept stages of algorithm development and testing is the MNIST dataset (and similar ones like the Fashion-MNIST, Kuzushiji-MNIST, and EMNIST). They have been well studied and their early-stage testing value is undoubtable, but when used in isolation they might offer a partial biased view of a model's capabilities. For instance, their grayscale characteristic (i.e., that they consist of single-channel images) conceals the fact that a substantial number of deep SOMs are not in a position to process and model colored images whereas a handful of advanced ones like [14–16,47] succeed in doing. Nevertheless, preliminary results for the MNIST benchmark of a pilot SOCOM study can be found in [48].

The more challenging STL-10 benchmark dataset [49] was used in this experimental setup. More specifically, STL-10 consists of colored 96 × 96 pixel images: 5000 labeled training images, 8000 labeled testing images, and 100,000 unlabeled images. This choice apart from testing SOCOM's performance on a more difficult dataset was also dictated by the need to work with and demonstrate NMVs with higher resolution images.

The visualization-oriented experiments involved SOCOMs using the vgg11 [50] as their backend architecture. Including the parameters of the neural output, the overall architectures have 12 layers of tunable weights. As is frequently the case in the literature [31,39,51,52], the vgg backend architecture was selected because the activation maximization results are visually more recognizable and better interpretable. Since Pytorch's vgg11 model is pre-trained on the Imagenet dataset images were resized to 224 × 224. Certain modifications have been carried out on the vgg11 so as to give rise to the final structure of the SOCOM. (1) The vgg11's last fully connected layer receiving 4096 inputs (feature values) and yielding 1000 activations alongside its complementary soft-max component were replaced by a hexagonal lattice of neurons. (2) A 1D pooling layer, receiving the weighted outputs of the lattice, has been added for facilitating the devised backpropagation optimization algorithm. Subsequently, exactly because SOCOM's construction provides this capability, transfer learning [53,54] was utilized for obtaining the initial weight/parameter values of the hidden layers that are shared with the vgg11 architecture. Having a far better starting point for the parameters' estimations in contrast to random initializations has definitely accelerated all the stages of the training procedures. The codebook parameters have been initialized according to the methodology described in [55], using a uniform distribution.

There is one more structural hyperparameter: the number of neurons in the output layer. A specific number for the neural map's (per row and column) dimensionalities is not crucial, actually, a wide range of grid sizes result in equally performing SOCOMs. As long as the total number of neurons remains above the number of data categories/labels, no evident deterioration is observed. Obviously, larger maps provide more space for representing intra-cluster homogeneity and wider margins for expressing inter-cluster distances, but this comes at the expense of additional computation time and of neurons being the best match for few or no samples. A characteristic 8 × 6 hexagonal array of neurons has been chosen for visualization reasons. The resultant network was trained/adapted by the SOCOM unsupervised training algorithm where modifications were allowed down to the first fully connected layer. This approach was followed so as to have a common set of identical features, stemming from the convolutional layers, between the SOCOM and the classification models, so as to be in a position to compare the corresponding activation maximization images of the classifier's output units and the NMV images of the cluster neurons.

Overall, the learning hyperparameter selection strategies that have been used could be summarized as follows. In accordance with the tenfold cross-validation technique, grid-based parameter configurations were evaluated/compared using purity as a performance measure. For certain hyperparameters like the learning rate, weight decay, and momentum, the upper and lower limits of their value ranges have been further refined according to the graph-based technique described in [56]. The top-performing (in certain cases by a large margin) SOCOM learning algorithms incorporated either the Adam or Adamax optimizers. It is interesting to note that this finding is in agreement with the observation that "Adam has been empirically shown to outperform most other optimizers in deep learning networks" [57]. Eventually, the neural output map's training hyperparameters that were selected were learning rate = 0.2 and weight decay = 0.001. The backend architecture's tuning hyperparameters were learning rate = $5 \cdot 10^{-5}$ and weight decay = 0.01. Sigma decreased linearly from 0.55 to 0.35, when it reached 0.35 it remained constant for the remaining duration of the training phase. Training batches comprised of 200 randomly selected samples and the total duration of the learning phase was set to 1000 steps. The reason for opting for bigger batch sizes is that they had a stabilizing effect on the learning curve by limiting fluctuations, and frequently, achieved a better performance overall.

In a similar fashion, a grid-based search was also conducted for finding a set of hyperparameters (learning rate, weight decay, momentum, adaptation steps), with respect to the optimization dictated by the (14) and (15) equations that produced visually recognizable images. In each independent run, the starting point was a colored image normalized around zero. Excluding extreme hyperparameter choices, hyperparameter values that complemented each other nearly always resulted in interpretable visualizations. One such indicative NMV is illustrated in Figure 6. The first expected, but at the same time important, point to make is that the images characterizing each neuron on the grid are in a one-to-one correspondence with the calculated cluster categories as these have been defined by posterior majority voting over the assigned training samples at each neuron.

Nearly always, the majority of each neuron's samples determine the contents of the produced image. It is also interesting to note that classes that share common visual characteristics like deer-horse, cat-dog, and truck-airplane reside in adjacent neurons/clusters, something that further supports the continuity characteristic of the SOCOM mappings.

Figure 7 contains juxtapositions between parts from the NMV that describe individual neurons of the SOCOM and the units of the vgg11 classifier that encode the same categories as the neurons do. By inspecting the respective pairs it can be seen that they are closely correlated in the sense that they focus on the same characteristics in each image category. Additionally, as expected, identical hyperparameter sets yield similar images with similar quality and comparable depicted information. From a more macroscopic point of view, one could notice that both networks seem to focus on the same aspects of the input samples in order to achieve the respective clustering and classification results. In the case of animals, these are mainly distinctive parts of the head whereas in the case of vehicles these are (oblong) straight or diagonal edges. The aforementioned remarks might seem trivial but when the different network output mechanisms are taken into consideration (affine transformations followed by a soft-max nonlinearity vs. competition based on weighted distances over a topologically arranged lattice) then the results provide additional proof in favor of the activation maximization technique, as far as the robustness and generality of its use are considered. Since both the SOCOM and the CNN classifier share the same backbone architecture, this argument could be extended to include the image modeling capabilities of CNNs.

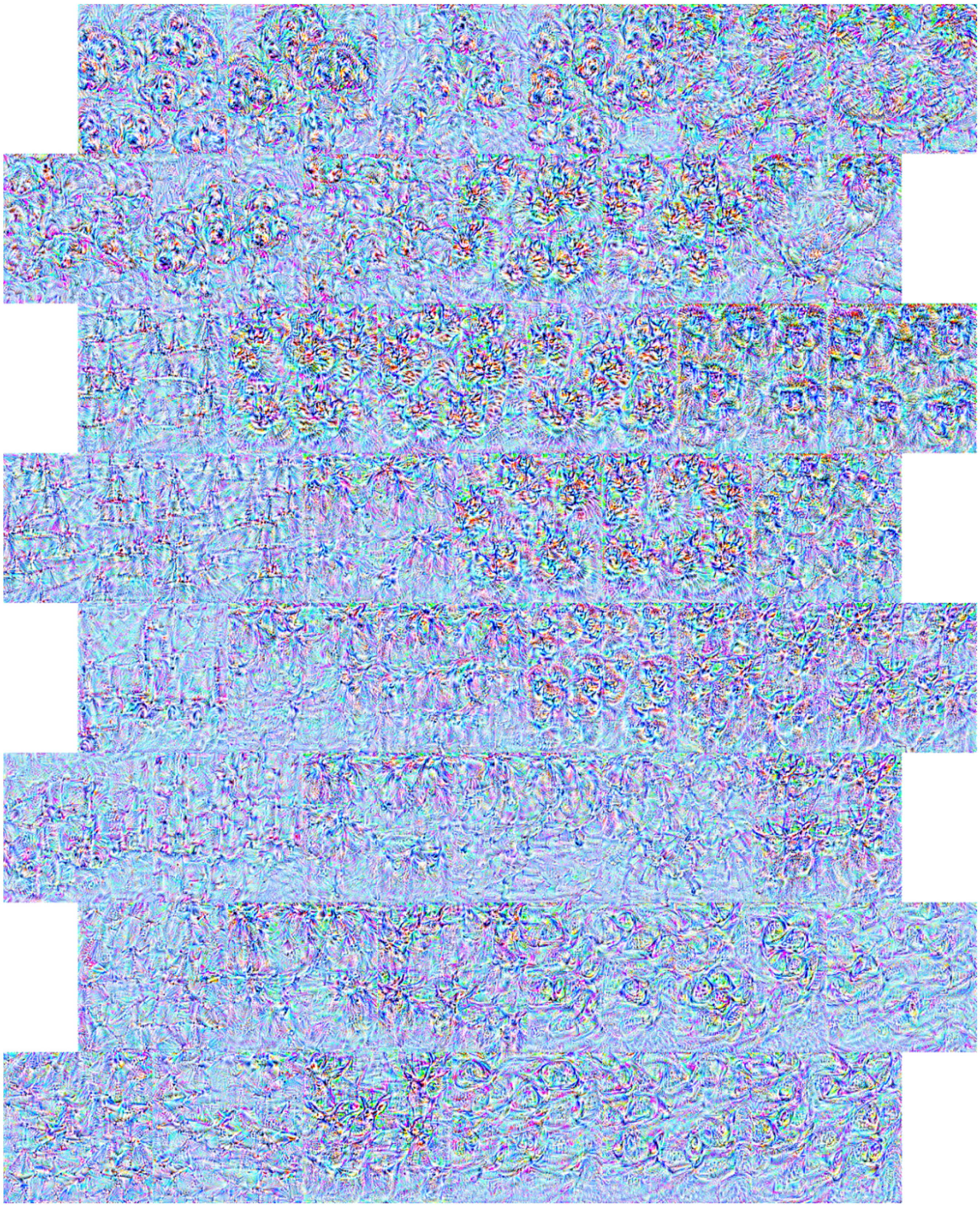

Figure 6. Neural map visualization (NMV) of the 8 × 6 neural output map of a SOCOM trained on the STL-10 benchmark dataset. Each individual neuron of the grid is represented by a synthetic image that depicts what the neuron models and which are the representations/patterns maximizing its response. As can be seen, there is a one-to-one correspondence between the

individual cluster/neuron visualizations and the respective categories obtained after posterior labeling of each neuron by applying the majority voting scheme. With respect to the topographical arrangement of the neural output map this posterior labeling is the following.

dog	dog	dog	dog	bird	bird
dog	dog	cat	cat	cat	bird
ship	cat	cat	cat	monkey	monkey
ship	ship	horse	cat	cat	monkey
truck	horse	horse	cat	deer	deer
truck	truck	horse	horse	horse	deer
airplane	horse	deer	unknown	unknown	car
airplane	airplane	deer	car	car	car

Best viewed in the electronic version of the paper, zoomed in.

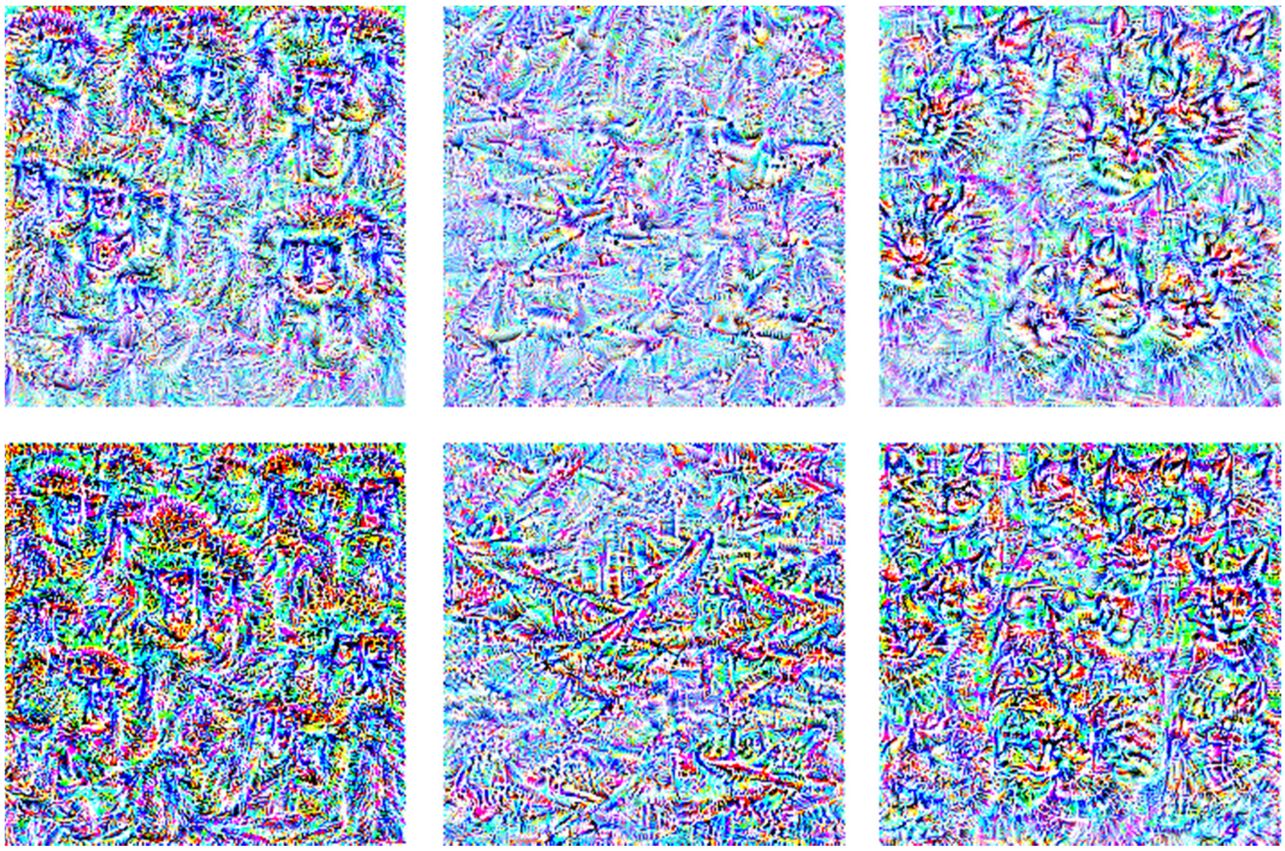

Figure 7. Upper row: selected synthetic images (taken from the overall NMV) of SOCOM neurons representing monkeys, airplanes, and cats. Lower row: the analogous synthetic images of the output units of the vgg11 classifier that represent the exact same categories of data. As can be seen, without being identical, they focus on the same characteristics and patterns of the input data (edges, parts of the head, vertices at different scales, and orientations) to achieve the respective clustering and classification results, despite the fact that the underlying mechanisms of their output layers are different.

The projection shown in Figure 8 has been constructed to further demonstrate SOCOM's clustering continuity and self-organizing capabilities. Essentially, the same 8×6 neural map is depicted, but in this case, each neuron is represented by the unique individual images, from the testing batch of the STL-10, that demonstrate the best/optimal fit with respect to the devised energy formula (viz. mapping schema). Apart from the evident correspondence between the synthetic images of Figure 6, the actual images of Figure 8, and the posterior labeling of the output neurons, an additional observation needs to be made. Not only do common/shared visual characteristics result in mappings that are adjacent on the SOCOM neural map but even more subtle differences like perspective, orientation, or focus are distinguished and are subsequently assigned to different but still neighboring neurons. For instance, the neurons clustering/describing cars (on the bottom right of the mapping) specialize accordingly in modeling either the side of the vehicle, its front-back, or close-up viewing points. It is also interesting to point out that the number of outliers is significantly low and the few ones that can be traced are located at the boundaries of their respective clusters (for instance between the neurons which model cats and dogs lying on the floor).

A key characteristic of the NMV that should be pointed out is that it does not require any type of class/category assignments, or posterior information in general, at any stage of its operation. This is fully aligned with the end-to-end unsupervised property of the SOCOM learning algorithm. The combination of these two algorithmic components of the SOCOM brings forth a network that is in a position to train, produce clusterings/mappings, and visualize them without using any type of label information at any stage of the whole procedure. The NMV offers an unsupervised visual interpretation of what the SOCOM models, or equivalently, a projection of the achieved higher-level representations onto the output neural map.

3.2. Quantitative Analysis

For evaluating the quality of the clustering output, and more specifically in the case of SOMs, the quality of the mapping output various internal and external criteria have been introduced. Internal criteria are more qualitative in the sense that they evaluate clustering results indirectly (e.g., by means of organization, compactness/sparseness, isolation, and preservation), whereas external are more quantitative since by measuring the match between clustering and external (e.g., human-based) categorizations they are in a position to provide more precise assessments. Despite the fact that in the general case there is no binding rule stating that class categorizations are in a one-to-one correspondence with potential cluster assignments, nearly always in the related literature the preferred criterion is purity; an external type criterion:

$$PUR = \frac{1}{S}\sum_{p=1}^{P}\max_{1\leq t\leq T}\left|s_p \cap s_t\right|. \tag{16}$$

The subscript p denotes the partitioning of a set of S samples into P distinct clusters (a posteriori estimated by the model); similarly, the subscript t denotes the assignment of these samples into T categories (a priori defined in the dataset). As expected, its resulting values lie in the $[0,\ 1]$ interval. Obviously, purity identifies with accuracy given that the majority voting principle is utilized for labeling each individual cluster. Although purity intuitively is rather straightforward/precise, it tends to favor small (in sample numbers) clusters like singletons.

Training trajectories showing the error (12) and the purity/accuracy (16) graphs from indicative top-performing SOCOM's are given in Figure 9. Two SOCOM models are depicted, a network initialized with transfer learning from a problem-specific vgg11 classifier (SOCOM-PSTL) and one without (SOCOM). As it is reasonable to expect, the former demonstrates better performance (both in terms of error and accuracy) in comparison to the latter whereby it also converges faster to a higher accuracy value. As can be observed, in either case, the coarse phase appears to last less than 15 epochs followed by the fine-tuning (viz. convergence) phase of the SOCOM learning procedure. It is also interesting to note that despite the fact that the SOCOM's error drops faster it does not reach the low values of SOCOM-PSTL; nevertheless, with respect to accuracy it manages to improve significantly later along in the training procedure.

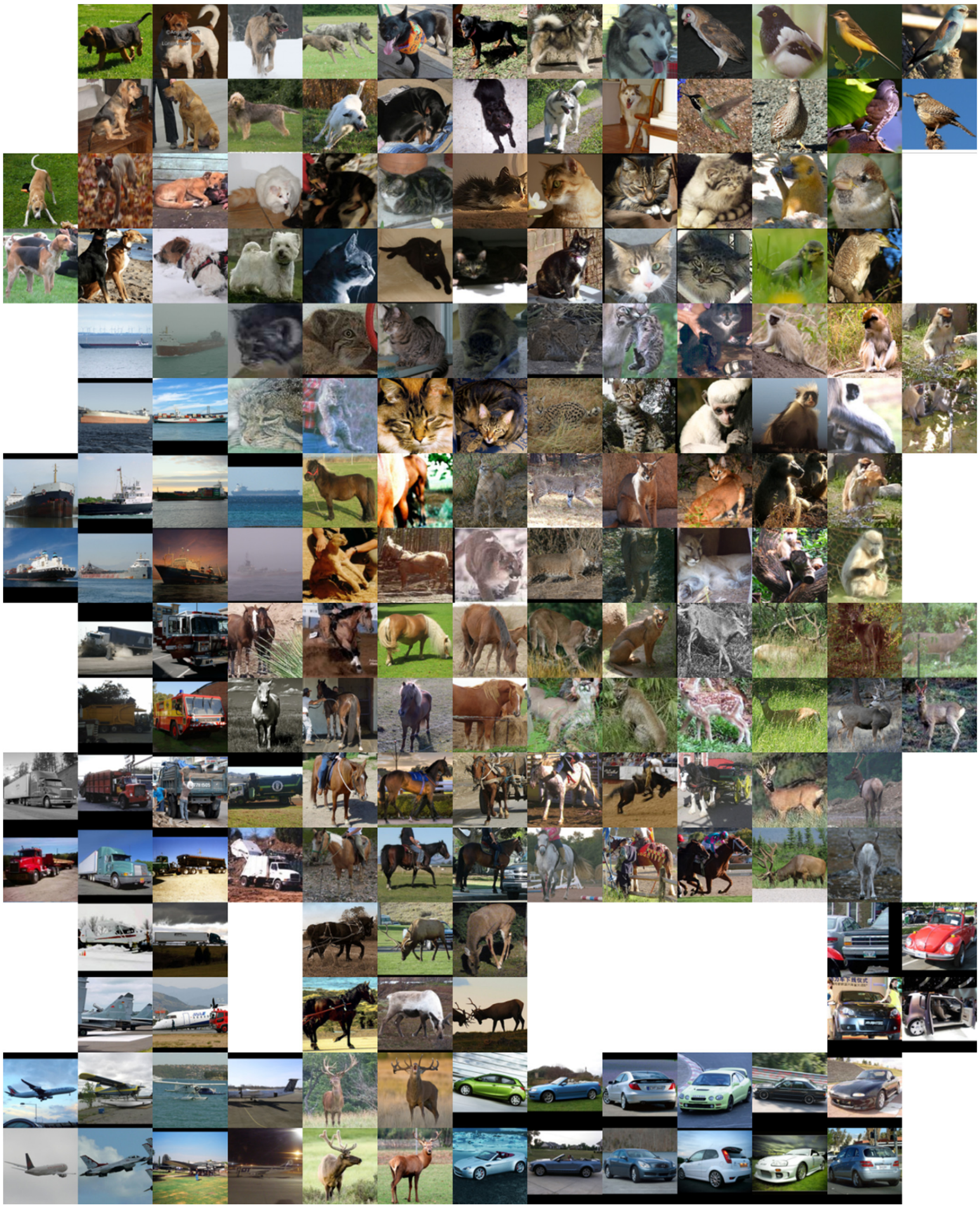

Figure 8. The best-matching images, from the STL-10 testing batch, of every neuron forming the 8 × 6 SOCOM neural map. Each individual neuron is represented by the four images yielding the highest activations among all the images assigned to the specific neuron. If a neuron describes/contains less than four images (or even no images at all) this is shown by empty/white slots.

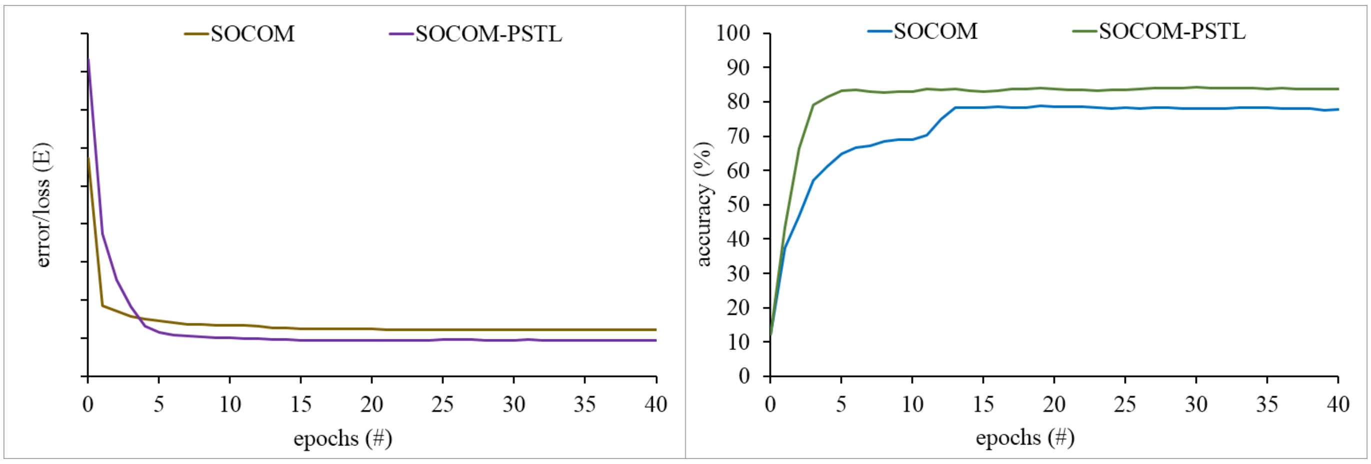

Figure 9. Training evolution of two characteristic types of SOCOMs, alongside the trajectories of the respective performance criteria. (**Left**) The networks' error/loss values across training time (i.e., epochs). (**Right**) The achieved accuracies at each stage of the unsupervised learning procedure.

The 8000 test images that have been ignored/excluded during training, were used for estimating SOCOM's accuracy. A list of top-performing (also in terms of accuracy) characteristic deep SOMs, (partially) unsupervised learning CNNs, and CNN clustering techniques is summarized in Table 1. As can be seen, SOCOM belongs to the top-performing group of algorithms. Moreover, differently from the rest of the top-performing models, it achieves the reported accuracy rate by following an end-to-end unsupervised approach throughout both its learning phase and clustering operations.

One needs to be clear from the beginning with regards to the key difference between obtaining accuracies with a posterior labeling of neurons (as is the case for IIC, ADC, DAC, DEC, and SOCOM) and obtaining accuracies with the addition of a supervised model/layer (like MLP, SVM or fully connected soft-max network). For instance "in this work we propose an evaluation procedure consisting of applying the result (the feature vector) in a classification system and comparing it to other classifiers under the same datasets" [59]. Deterministically, the supervised layer approaches' results are expected to be higher-better since the unsupervised networks' outputs are treated as input features to a supervised network (which is obviously trained in a supervised manner). This type of experimental testing does reveal characteristics of the unsupervised module's output feature space but it offers an over-optimistic view of the network's clustering capabilities and performance. The end-to-end unsupervised learning networks that resort to this kind of feature space validation are all those scoring above 66%; this fact renders SOCOM as the only algorithm in the group capable of producing clustering and (indirectly through neuron posterior labeling) classification results without the requirement of a front-end output supervised layer. Actually, under a puristic unsupervised learning comparison, the models that utilize class-label information at whichever part of their operation should be excluded. Strictly speaking, this exclusion would also involve SOCOM-PSTL since it is initialized with transfer learning from a problem-specific supervised classifier. The only algorithms conforming to such strict unsupervised learning and operating criteria are the SOCOM, IIC, ADC, DAC, and DEC. With respect to this experimental comparison setup, the SOCOM outperforms the rest by at least 18%.

The main objective of the present experimental series was to set in motion SOCOM's algorithmic mechanisms in an effort to tangibly demonstrate and verify its capabilities and clustering performance. The primary goal of the reported results was to complement the theoretical merits of the proposed model with their practical applications.

Table 1. The reported accuracies of deep SOMs, (partially) unsupervised learning CNNs, and CNN clustering techniques on the STL-10 dataset. If a methodology necessitates an additional supervised training layer applied to its features for producing the reported results then this is specifically indicated in the last column and the exact types of the supervised layers are shown in the parentheses.

Model/Network	**Accuracy (%)**	**End-to-End Unsupervised Learning**	**Unsupervised Clustering and Classification Operations**
SOCOM-PSTL	84.19	●	●
Spatial Contrasting Initialization (Soft-max classifier) [58]	81.34	●	—
UDSOM (SVM classifier) [59]	80.19	●	—
SOCOM	78.7	●	●
Exemplar CNN (SVM classifier) [60]	74.2	●	—
Convolutional k-Means Clustering (Linear classifier) [61]	74.1	●	—
Zero-bias CNN ADCU (Soft-max classifier) [62]	70.2	●	—
MSRV+C-SVDDNet (SVM and soft-max classifier) [63]	68.23	●	—
Committees of Deep Networks (SVM classifier) [64]	68.0	●	—
Unsupervised Feature Learning by Augmenting Single Images (SVM classifier) [65]	67.4	●	—
Hierarchical Matching Pursuit (SVM classifier) [66]	64.5	●	—
Discriminative Convolution with Fisher Weight Map (Logistic regression classifier) [67]	66.0	●	—
IIC [68]	59.8	●	●
ADC [69]	53.0	●	●
DAC [70]	47.0	●	●
DEC [71]	35.9	●	●

4. Conclusions

The SOCOM prototype is in a position, in theory and in practice, to incorporate deep convolutional networks and to train them with a gradient backpropagation algorithm specifically tailored to meet the requirements of the architectures' complexity, depth, and parameter size. The construction of the SOCOM intrinsically offers the capability to make use of transfer learning from pre-trained CNNs. Furthermore, the low-dimensional spatially ordered array of output neurons, which is overlaid above the embedded hidden layer features/representations of multi-channel inputs (e.g., colored images or sequences of images/signals), provides topology-driven clusterings and visualizations. In particular, the devised unsupervised learning visualization technique apart from offering insight and interpretation of the SOCOM's clustering operation and neural mapping also provides defensible indications regarding the formation of higher representations that comprise low-level distributed partial features.

Finally, it is reasonable to expect that the present self-contained study of the SOCOM prototype could give rise to a number of closely related research directions pointing towards enriching and diversifying the model, and towards promoting accessibility and ease-of-use of the SOCOM variants to the scientific research community. We believe that promising research paths to follow have been identified. We are undertaking certain parts of this research work, which we will make publicly available in the nearest future.

Author Contributions: Conceptualization, C.F., Y.P., S.P.S. and S.A.M.; formal analysis, C.F. and Y.P.; investigation, C.F. and Y.P.; methodology, C.F., Y.P., S.P.S. and S.A.M.; project administration, S.A.M.; software, C.F. and Y.P.; supervision, S.P.S. and S.A.M.; validation, C.F. and Y.P.; visualization, C.F. and Y.P.; writing—original draft, C.F. and S.A.M.; writing—review and editing, Y.P. and S.P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014–2020" in the context of the project "Self-Organizing Convolutional Maps" (MIS 5050185).

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
2. Northcutt, C.G.; Athalye, A.; Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv* **2021**, arXiv:2103.14749.
3. Malondkar, A.; Corizzo, R.; Kiringa, I.; Ceci, M.; Japkowicz, N. Spark-GHSOM: Growing hierarchical self-organizing map for large scale mixed attribute datasets. *Inf. Sci.* **2019**, *496*, 572–591. [CrossRef]
4. Forti, A.; Foresti, G.L. Growing Hierarchical Tree SOM: An unsupervised neural network with dynamic topology. *Neural Netw.* **2006**, *19*, 1568–1580. [CrossRef]
5. Jin, H.; Shum, W.-H.; Leung, K.-S.; Wong, M.-L. Expanding self-organizing map for data visualization and cluster analysis. *Inf. Sci.* **2004**, *163*, 157–173. [CrossRef]
6. Hsu, A.L.; Tang, S.-L.; Halgamuge, S.K. An unsupervised hierarchical dynamic self-organizing approach to cancer class discovery and marker gene identification in microarray data. *Bioinformatics* **2003**, *19*, 2131–2140. [CrossRef]
7. Lawrence, S.; Giles, C.L.; Tsoi, A.C.; Back, A.D. Face recognition: A convolutional neural-network approach. *IEEE Trans. Neural Netw.* **1997**, *8*, 98–113. [CrossRef]
8. Liu, N.; Wang, J.; Gong, Y. Deep self-organizing map for visual classification. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015.
9. Hankins, R.; Peng, Y.; Yin, H. Towards complex features: Competitive receptive fields in unsupervised deep networks. In Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning, Madrid, Spain, 21–23 November 2018.
10. Wickramasinghe, C.S.; Amarasinghe, K.; Manic, M. Deep self-organizing maps for unsupervised image classification. *IEEE Trans. Ind. Inform.* **2019**, *15*, 5837–5845. [CrossRef]
11. Aly, S.; Almotairi, S. Deep convolutional self-organizing map network for robust handwritten digit recognition. *IEEE Access* **2020**, *8*, 107035–107045. [CrossRef]
12. Friedlander, D. Pattern Analysis with Layered Self-Organizing Maps. *arXiv* **2018**, arXiv:1803.08996.
13. Pesteie, M.; Abolmaesumi, P.; Rohling, R. Deep neural maps. *arXiv* **2018**, arXiv:1810.07291.
14. Stuhr, B.; Brauer, J. Csnns: Unsupervised, backpropagation-free convolutional neural networks for representation learning. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019.
15. Part, J.L.; Lemon, O. Incremental on-line learning of object classes using a combination of self-organizing incremental neural networks and deep convolutional neural networks. In Proceedings of the Workshop on Bio-Inspired Social Robot Learning in Home Scenarios (IROS), Daejeon, Korea, 9–14 October 2016.
16. Wang, M.; Zhou, W.; Tian, Q.; Pu, J.; Li, H. Deep supervised quantization by self-organizing map. In Proceedings of the 25th ACM international conference on Multimedia, Mountain View, CA, USA, 23–27 October 2017.
17. Kohonen, T. *Self-Organizing Maps*; Springer: Berlin/Heidelberg, Germany; New York, NY, USA, 1995.
18. Heskes, T. Energy functions for self-organizing maps. In *Kohonen Maps*; Elsevier: Amsterdam, The Netherlands, 1999; pp. 303–315.

19. Ferles, C.; Stafylopatis, A. Self-organizing hidden markov model map (SOHMMM). *Neural Netw.* **2013**, *48*, 133–147. [CrossRef] [PubMed]
20. Ferles, C.; Papanikolaou, Y.; Naidoo, K.J. Denoising autoencoder self-organizing map (DASOM). *Neural Netw.* **2018**, *105*, 112–131. [CrossRef]
21. Ultsch, A. Maps for the visualization of high-dimensional data spaces. In Proceedings of the Workshop on Self Organizing Maps, Hibikono, Japan, 11–14 September 2003.
22. Ultsch, A. Clustering with SOM: Uˆ* C. In Proceedings of the Workshop on Self-Organizing Maps, Paris, France, 5–8 September 2005.
23. Ferles, C.; Beaufort, W.-S.; Ferle, V. Self-Organizing Hidden Markov Model Map (SOHMMM): Biological sequence clustering and cluster visualization. In *Hidden Markov Models*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 83–101.
24. Tasdemir, K.; Merényi, E. Exploiting data topology in visualization and clustering of self-organizing maps. *IEEE Trans. Neural Netw.* **2009**, *20*, 549–562. [CrossRef] [PubMed]
25. Brugger, D.; Bogdan, M.; Rosenstiel, W. Automatic cluster detection in Kohonen's SOM. *IEEE Trans. Neural Netw.* **2008**, *19*, 442–459. [CrossRef] [PubMed]
26. Pölzlbauer, G.; Dittenbach, M.; Rauber, A. Advanced visualization of self-organizing maps with vector fields. *Neural Netw.* **2006**, *19*, 911–922. [CrossRef] [PubMed]
27. Yin, H. ViSOM-a novel method for multivariate data projection and structure visualization. *IEEE Trans. Neural Netw.* **2002**, *13*, 237–243.
28. Pampalk, E.; Rauber, A.; Merkl, D. Using smoothed data histograms for cluster visualization in self-organizing maps. In Proceedings of the International Conference on Artificial Neural Networks, Madrid, Spain, 28–30 August 2002.
29. Vesanto, J. SOM-based data visualization methods. *Intell. Data Anal.* **1999**, *3*, 111–126. [CrossRef]
30. Erhan, D.; Bengio, Y.; Courville, A.; Vincent, P. Visualizing higher-layer features of a deep network. *Univ. Montr.* **2009**, *1341*, 1.
31. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv* **2013**, arXiv:1312.6034.
32. Nguyen, A.; Yosinski, J.; Clune, J. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv* **2016**, arXiv:1602.03616.
33. Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T.; Lipson, H. Understanding neural networks through deep visualization. *arXiv* **2015**, arXiv:1506.06579.
34. Mahendran, A.; Vedaldi, A. Understanding deep image representations by inverting them. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–25 June 2005.
35. Olah, C.; Mordvintsev, A.; Schubert, L. Feature visualization. *Distill* **2017**, *2*, e7. [CrossRef]
36. Olah, C.; Satyanarayan, A.; Johnson, I.; Carter, S.; Schubert, L.; Ye, K.; Mordvintsev, A. The building blocks of interpretability. *Distill* **2018**, *3*, e10. [CrossRef]
37. Carter, S.; Armstrong, Z.; Schubert, L.; Johnson, I.; Olah, C. Activation atlas. *Distill* **2019**, *4*, e15. [CrossRef]
38. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zürich, Switzerland; 2014.
39. Zintgraf, L.M.; Cohen, T.S.; Welling, M. A new method to visualize deep neural networks. *arXiv* **2016**, arXiv:1603.02518.
40. Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* **2015**, *10*, e0130140. [CrossRef] [PubMed]
41. Nguyen, A.; Yosinski, J.; Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 12 2015.
42. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2014**, arXiv:1412.6572.
43. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2013**, arXiv:1312.6199.
44. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
45. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv* **2017**, arXiv:1711.05101.
46. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G. On the importance of initialization and momentum in deep learning. In Proceedings of the International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013.
47. Braga, P.H.; Medeiros, H.R.; Bassani, H.F. Deep Categorization with Semi-Supervised Self-Organizing Maps. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020.
48. Ferles, C.; Papanikolaou, Y.; Savaidis, S.P.; Mitilineos, S.A. Deep learning self-organizing map of convolutional layers. In Proceedings of the 2nd International Conference on Artificial Intelligence and Big Data (AIBD 2021), Vienna, Austria, 20–21 March 2021; pp. 25–32.
49. Coates, A.; Ng, A.; Lee, H. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011. JMLR Workshop and Conference Proceedings.
50. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
51. Yu, W.; Yang, K.; Bai, Y.; Xiao, T.; Yao, H.; Rui, Y. Visualizing and comparing AlexNet and VGG using deconvolutional layers. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.

52. Nam, W.-J.; Choi, J.; Lee, S.-W. Interpreting Deep Neural Networks with Relative Sectional Propagation by Analyzing Comparative Gradients and Hostile Activations. *arXiv* **2020**, arXiv:2012.03434.
53. Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 28 June 2014; pp. 806–813.
54. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? *arXiv* **2014**, arXiv:1411.1792.
55. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings.
56. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE winter conference on applications of computer vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017.
57. Pointer, I. *Programming PyTorch for Deep Learning: Creating and Deploying Deep Learning Applications*; O'Reilly Media, Inc.: Newton, MA, USA, 2019.
58. Hoffer, E.; Hubara, I.; Ailon, N. Deep unsupervised learning through spatial contrasting. *arXiv* **2016**, arXiv:1610.00243.
59. Sakkari, M.; Zaied, M. A Convolutional Deep Self-Organizing Map Feature extraction for machine learning. *Multimed. Tools Appl.* **2020**, *79*, 19451–19470. [CrossRef]
60. Dosovitskiy, A.; Fischer, P.; Springenberg, J.T.; Riedmiller, M.; Brox, T. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 1734–1747. [CrossRef]
61. Dundar, A.; Jin, J.; Culurciello, E. Convolutional clustering for unsupervised learning. *arXiv* **2015**, arXiv:1511.06241.
62. Paine, T.L.; Khorrami, P.; Han, W.; Huang, T.S. An analysis of unsupervised pre-training in light of recent advances. *arXiv* **2014**, arXiv:1412.6597.
63. Wang, D.; Tan, X. Unsupervised feature learning with C-SVDDNet. *Pattern Recognit.* **2016**, *60*, 473–485. [CrossRef]
64. Miclut, B. Committees of deep feedforward networks trained with few data. In Proceedings of the German Conference on Pattern Recognition, Münster, Germany, 2–5 September 2014.
65. Dosovitskiy, A.; Springenberg, J.; Brox, T. Unsupervised feature learning by augmenting single images. *arXiv* **2014**, arXiv:1312.5242.
66. Bo, L.; Ren, X.; Fox, D. Unsupervised feature learning for RGB-D based object recognition. In *Experimental Robotics*; Springer: Berlin/Heidelberg, Germany, 2013.
67. Nakayama, H. Efficient Discriminative Convolution Using Fisher Weight Map. In Proceedings of the BMVC, Bristol, UK, 9–13 September 2013.
68. Ji, X.; Henriques, J.F.; Vedaldi, A. Invariant information clustering for unsupervised image classification and segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27–28 October 2019.
69. Haeusser, P.; Plapp, J.; Golkov, V.; Aljalbout, E.; Cremers, D. Associative deep clustering: Training a classification network with no labels. In Proceedings of the German Conference on Pattern Recognition, Stuttgart, Germany, 9–12 October 2018.
70. Chang, J.; Wang, L.; Meng, G.; Xiang, S.; Pan, C. Deep adaptive image clustering. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
71. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised deep embedding for clustering analysis. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.