Deep Self-Organizing Maps for Unsupervised Image Classification

Chathurika S. Wickramasinghe, Kasun Amarasinghe, *Student Member, IEEE,* and Milos Manic, *Senior Member, IEEE*

Abstract—**The Deep Self-Organizing Map (DSOM) was introduced to embed hierarchical feature abstraction capability to SOMs. This paper presents an extended version of the original DSOM algorithm (E-DSOM). E-DSOM enhances the DSOM in two ways: 1) learning algorithm is modified to be completely unsupervised, and 2) architecture is modified to learn features of different resolution in hidden layers. E-DSOM has three main advantages over the original DSOM: 1) improved classification accuracy, 2) improved generalization capability and 3) need of fewer sequential layers (reduced training time). E-DSOM was tested on benchmark and real-world datasets and was compared against DSOM, SOM, Stacked Autoencoder (AE), and Stacked Convolutional Autoencoder (CAE). Experimental results showed that the E-DSOM outperformed DSOM with improvements of classification accuracy up to 15% while saving training time up to 19% on all datasets. Moreover, E-DSOM evidenced better generalization capability compared to the DSOM by showing superior performance on all datasets with induced noise. Further, E-DSOM showed comparable performance to the AE and the CAE while outperforming them on two datasets.**

Index Terms—**Deep Self Organizing Maps; Unsupervised Learning; Image classification; Deep Learning; Generalization**

I. Introduction

The ubiquity of image sensing devices has led to the generation of substantial amounts of image data in real-world industrial settings[1]. In addition to the directly acquired images, recently it has been shown that numerical data acquired from industrial processes can be viewed as images so that image data mining techniques can be applied on them [1], [2]. Successful mining of these images (classification, clustering) can lead to several advantages including process optimization, fault diagnosis, and improved cyber-security [1], [3], [4], [5], [6], [7].

In image classification, deep learning algorithms such as Deep Convolutional Neural Networks (CNN) have shown unprecedented performance [8], [9]. Recent attempts have focused on improving the efficiency of these algorithms by developing light-weight deep neural networks [10]. Despite many advantages, one major drawback of these state-of-the-art image classification algorithms is that they are dependent on the availability of large labeled datasets. The scarcity of labeled data in the real world is a major hurdle to deploy supervised models in the real-world [3], [11], [12], [13]. Therefore, unsupervised approaches are ideal to leverage the abundantly available unlabeled data in industrial applications [3], [12], [13].

Several unsupervised image classification methodologies have been explored in literature such as Bayesian hierarchical clustering [14], [15] and Markovian models [16]. In more recent attempts, specialized deep learning methodologies such as Spiking Neural Networks (SNN) [17] and Generative Adversarial Nets (GANs) [18] were proposed. These algorithms have shown comparable performance with supervised algorithms for the MNIST dataset [17]. However, these models have some limitations when it comes to deploying them in the real-world. For example, the complex architecture of SNNs leads to low understandability and requirements of specialized hardware to deploy [19], [20]. Similarly, GANs suffer from poor interpretability and has been shown that they suffer from high training times [21], [22]. In addition, deep learning methodologies such as stacked convolutional Autoencoders (CAEs) have shown much promise in unsupervised learning for image classification [9], [23]. In this work, we focus on using Self-Organizing Maps (SOMs) based methodology for image classification using unsupervised deep learning.

The SOM is an unsupervised learning architecture capable of mapping high-dimensional data distributions onto low-dimensional distributions while preserving the most important topological relationships of input data. Therefore, SOMs are suitable for visualizing and mining high dimensional data [24], [25], [26]. Other advantages of SOMs include interpretability and understandability [23], [27], ease of optimization [28], and better capability of revealing overlapping structures in clusters compared to traditional clustering methods [29]. SOMs have been successful in a multitude of areas including speech recognition, robotics and process control [27], [30], [31], [2], [3], [4], [5], [6], [7], [8]. The major drawback of SOMs is its limited capability of high-level feature abstraction due to the shallow structure [32].

One of the recent attempts at alleviating this limitation was to explore a deep architecture of SOMs, named Deep Self-Organizing Maps (DSOM) by Liu et al [25]. Authors tested the DSOM on the MNIST dataset and were able to achieve a better classification accuracy compared to the single layer SOM. Since DSOM architecture uses the same learning mechanism as SOMs, it inherits all the advantages of SOMs mentioned above. However, the authors of [25] explored a supervised learning algorithm with DSOM and thus relied on the availability of labeled data. In our previous work, we explored a parallelized version of an unsupervised DSOM. It showed promising results on the MNIST dataset [33]. An

Chathurika S. Wickramasinghe, Kasun Amerasinghe, and Milos Manic are with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: brahmanacsw@vcu.edu, amarasinghek@vcu.edu, misko@ieee.org).

accurate unsupervised DSOM architecture has the following main advantages: 1) the ability to leverage unlabeled datasets, 2) hierarchical feature abstraction based unsupervised learning, and 3) the ability to deploy without special hardware.

This paper presents an extended architecture of the initially proposed DSOM (E-DSOM). E-DSOM enhances the DSOM in two ways: 1) the learning algorithm is completely unsupervised and 2) the architecture learns features of different resolutions in parallel in a single hidden layer. Please note that in the rest of the paper, DSOM refers to the architecture proposed by Liu et al. in [25] and E-DSOM refers to the architecture presented in this paper. The main contributions of this work are summarized as follows:

1) An unsupervised, easy to understand, easy to implement deep SOM architecture for image classification. The goal of this work is not to improve on the accuracies of other supervised learning architectures such as CNN. The goal is to present an unsupervised learning methodology, with high-level feature abstraction capability for image classification.
2) A deep SOM architecture capable of learning features of different resolutions simultaneously. We hypothesize that this capability improves classification accuracy and the generalization capability of the model. Further, we hypothesize that this will result in a shallower model compared to the DSOM and lead to reduced training times.

The rest of the paper is organized as follows. Section II provides the background which includes SOM and DSOM algorithms; Section III presents the E-DSOM algorithm; Section IV presents the experimental setup and results, and finally, Section V concludes the paper and discusses future directions.

II. Background

This section provides the background information needed to present the E-DSOM algorithm. First, the single-layered SOM is introduced. Then, the DSOM algorithm proposed by Liu et al. in [25] is presented.

A. Self-Organizing Maps

SOMs consist of a topological neuron grid (typically 2D) with each neuron consisting of a weight vector. The SOM adapts itself to the topological properties of input data using the unsupervised "winner-take-all" learning algorithm. Both DSOM and E-DSOM use this as the underlying learning mechanism in the hidden layers.

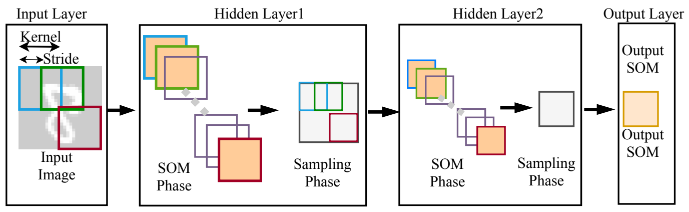

Fig. 1. Two layered DSOM architecture used for handwritten character recognition [25]

TABLE I
Algorithm for training the Self-Organizing Map

```
Algorithm I: SOM Training
Inputs: Training set of images ($X$)
Outputs: Trained SOM
1:  Random Weight initialization
2:  for each epoch e do
3:      for number of training samples do
4:          $x \leftarrow$ pick random input record from X
5:          $md \leftarrow$ initialize to the largest float
6:          for number of neurons in SOM do
7:              $d_i \leftarrow \|X - W_i\|$
                % find the BMU
8:              if $d_i < md$ do
9:                  $BMU_x \leftarrow W_i$ % weight of BMU
10:                 $BMUIndex_x \leftarrow i$ % index of BMU
11:                 $md \leftarrow d_i$
12:             end if
13:         end for
            % update weights
14:         for number of neurons in SOM neighborhood do
15:             $n \longleftarrow e^{-\left(\frac{BMU_x - w}{2\delta t^2}\right)}$
16:             $\triangle W_i \longleftarrow W_i \times \alpha \times \eta \times (x - w)$
17:             $W_i \longleftarrow W_i + \triangle W_i$
18:         end for
            % Decay the neighborhood and learning rate
19:     end for
20: end for
```

The learning algorithm for a SOM is given in Algorithm I (Table I). For each input pattern, the SOM selects the neuron that best matches the pattern in terms of Euclidean distance. This neuron is called the Best Matching Unit (BMU). Then, the SOM updates weights of the neurons in the neighborhood of the BMU so that they move closer to the BMU (line 14-18 in **Algorithm I**). The learning rate and the radius of the BMU neighborhood are used as the controlling hyper-parameters.

The learning rate and the neighborhood radius is decayed with time. The neighborhood radius is halved at each epoch.

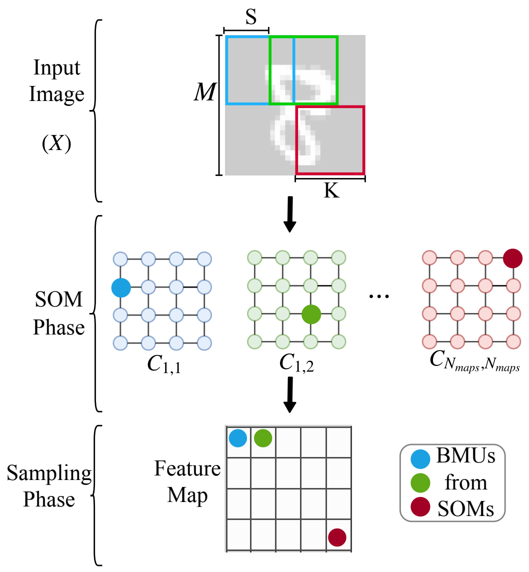

Fig. 2. Sampling layer creation in DSOM

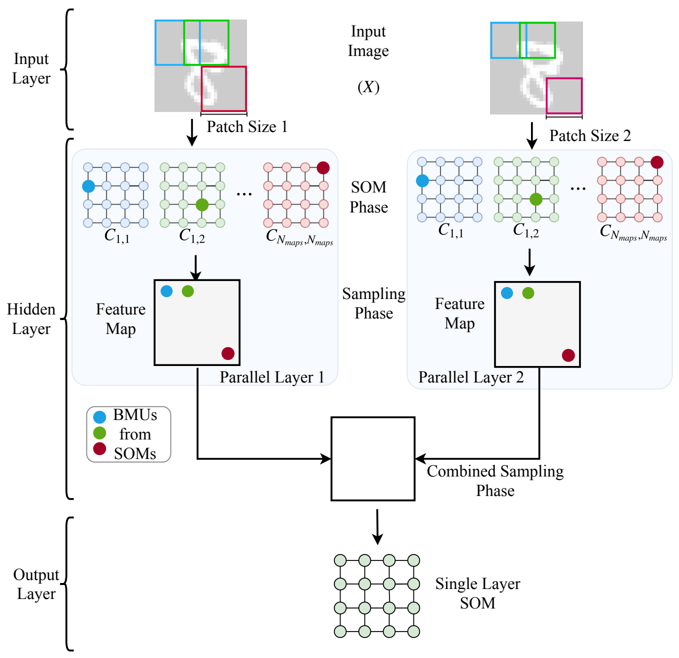

Fig. 3. E-DSOM architecture with one hidden layer (two parallel layers in the hidden layer)

The learning rate is decayed as follows:

$$\eta(t) = 0.49\left(1 - \frac{e}{epochs}\right) + 0.01 \tag{1}$$

Where e is the current epoch and $epochs$ is the total number of epochs. The most important hyper-parameter of the SOM is the size of the map. If the map size is too small, it will lead to the model not capturing the feature space adequately (under-fitting). Conversely, if the map size is too large, it will lead to over-fitting the training data and unnecessary computations.

B. Deep Self-Organizing Maps

The DSOM is a multi-layered SOM architecture, which consists of an input layer, hidden layers, and an output layer. The initial design of DSOMs merged the concepts of SOMs and Convolution Neural Networks (CNNs). SOMs provided the underlying learning mechanism to DSOM while CNNs inspired the high-level feature abstraction process.

In CNNs, in each hidden layer, each unit (neuron) receives inputs from a subset of units in the preceding layer (local receptive field/patch) [34], [35]. The lower-level features learned in the preceding layer are combined in the current hidden layer to generate higher-level features. This idea was incorporated into the DSOM architecture so that higher-level layers are capable of learning more abstract information than lower-level layers.

Although DSOM uses local receptive fields, the function of hidden layers is completely different from CNN. In CNN hidden layers, a convolution operation followed by a pooling step generates the feature map for the next layer. Conversely, hidden layers in DSOM process the patches with SOMs and aggregate the BMUs to generate the feature map. Therefore, the only similarity between DSOM and CNN is the notion of the local receptive field. Figure 1 shows a DSOM architecture with two hidden layers, which was used for MNIST classification by Liu et al [25].

The function of each layer in the DSOM can be summarized as follows:

Input layer: Forwards the input images to the DSOM

Hidden Layer: Each hidden layer consists of two phases: 1) SOM phase and 2) sampling phase. In the SOM phase, each input image is segmented into smaller local regions (patches). Then, each patch is sent to its own SOM unit in this layer, i.e. each patch is processed by its own SOM. Each SOM finds the BMU for the input patch using the Algorithm I. In the sampling phase, the BMUs of the hidden SOM units are combined to generate a single 2D grid (see Figure 2). This 2D grid acts as the input image (feature map) to the next hidden layer. This process is repeated for all hidden layers.

Output layer: The output layer contains a single SOM. It receives the feature map generated by the last hidden layer. The output SOM extracts abstract and pertinent information for classification.

III. Extended Deep Self-Organizing Maps

This section presents the E-DSOM architecture, its unsupervised learning algorithm, classifier implementation and a discussion on computational complexity.

A. E-DSOM Architecture

Similar to the DSOM architecture, the E-DSOM consists of an input layer, hidden layers, and an output layer. The main differences in the E-DSOM architecture are in the hidden layers. As opposed to the DSOM, E-DSOM hidden layers contain several parallel layers (See Figure 3). Each parallel layer has its SOM phase and sampling phase. In the sampling phase, first, a feature map for each parallel layer is created. Then, those feature maps are combined to generate one feature map.

E-DSOM uses different sized patches (multi-scale patches) in parallel SOM layers of the hidden layers. It has been shown that multi-scale patch approaches help to improve classification accuracy by extracting complementary information [36], [37]. Figure 3 shows an E-DSOM architecture with two parallel layers with different patch sizes.

The above architecture modification enables the algorithm to learn feature spaces of different size and resolution by using different map sizes and patch sizes in the parallel layers. We hypothesize that this ability will 1) improve classification accuracy, 2) improve generalization capability, and 3) reduce the need for sequential hidden layers (reduce training time). In this work, more emphasis is laid on changing the patch size, i.e., changing the size of the local region of focus to enable the learning features of different resolutions.

B. Training the E-DSOM

Training algorithm of the E-DSOM is presented in Algorithm II (Table II). Similar to the SOM and DSOM, weights of the network are randomly initialized. In a hidden layer,

TABLE II
ALGORITHM FOR TRAINING THE E-DSOM

```
Algorithm II: E-DSOM Training
Inputs: Training set of images ($X$)
Outputs: Trained E-DSOM
1:  Random Weight initialization
2:  for each epoch e do
3:      for number of training samples do
4:          $x \leftarrow$ pick random input record from X
5:          for each hidden layer l do
6:              $featureMapList \leftarrow empty\ list\ of\ length\ P$
7:              for each parallel SOM layer $p$ do
8:                  $featureMapList[p] \leftarrow ParallelLayer(x)$
9:              end for
10:             $x \leftarrow CombinedSampling(featureMapList)$
11:         end for
12:         OutputSOM $\leftarrow$ Algorithm I ( $x$ )
            % Find the BMU for ( $x$ ) using SOM algorithm
13:     end for
14: end for
```

```
Procedure I: ParallelLayer
Inputs: Input record ($x$), Number of patches ($p$)
Outputs: Sampled $featureMap$
1:  $featureMap \leftarrow$ empty list of length $p$
2:  for each patch $x^{‘}$ do:
3:      $index_x \leftarrow$ the location of $x^{‘}$ w.r.t. $x$
4:      $BMU_{x^{‘}} \leftarrow$ get BMU index for $x^{‘}$ on corresponding $SOM_{x^{‘}}$
5:      $featureMap[index] \leftarrow BMU_index$
6:  end for
```

```
Procedure II: CombinedSampling
Inputs: List of feature maps from each parallel layer ( $featureMapList$ )
Outputs: Combined Feature Map
1:  $comFeatureMap \leftarrow$ Append $featureMapList$ to a single list
2:  $l \leftarrow$ length of $comFeatureMap$
3:  if $\sqrt{l} \notin \mathcal{N}$ for;    $\mathcal{N} = \{1, 2, 3, 4, ...\}$
4:      Use zero-padding on $comFeatureMap$ until $\sqrt{l} \in \mathcal{N}$
5:  $CFM \leftarrow$ Reshape $comFeatureMap$ to a 2D vector of size $\sqrt{l} * \sqrt{l}$
6:  return CFM
```

the SOM phase consists of P parallel SOM layers with P different patch sizes (Algorithm II, lines 7-9). For each patch size, the number of patches along one dimension is calculated as follows:

$$N_{map} = ceil\Big(\frac{M-K}{S}\Big) + 1 \tag{2}$$

where $ceil(\cdot)$ calculates the smallest integer upper, M is the pixel width/height of the input image X ($M \times M$ image) K is the width/height of the patch ($K \times K$ patch) and S is the stride of the patch. Therefore, $N_{map} \times N_{map}$ number of patches are created form the input image for each patch size (see Figure 2), i.e. $N_{map} \times N_{map}$ number of SOMs are created for each parallel layer.

In all the parallel SOM layers, the BMU selection for its respective patches is carried out followed by its sampling process(see Procedure I). Therefore, P feature maps are created (see Algorithm II) [25]. All feature maps are converted to one-dimensional arrays and concatenated into a single array. Then, the resultant array is reshaped to a 2D grid which acts as the input image to the next hidden layer (see Procedure II).

After processing the hidden layers, the combined feature map generated from the last hidden layer acts as the input to the output SOM. The output layer SOM is trained using Algorithm I.

C. Classifier

A classifier is implemented based on the trained output layer SOM to assign the class labels to the input records. It has to be noted that E-DSOM algorithm is trained purely unsupervised, without using any prior knowledge about class labels.

The classifier requires some labeled data. Each neuron in the output layer SOM is assigned a class label using the labeled dataset. First, the labeled dataset is processed through the trained E-DSOM. For each neuron j in the output layer, the number of times it was selected as the BMU for each class is stored as, $N^{j,c}_{BMU}$ where c is the class label. The class label with the highest BMU frequency is assigned as the neuron label:

$$label_j = \underset{c}{\operatorname{argmax}}\, N^{j,c}_{BMU} \tag{3}$$

In case of a tie one of the tied labels of maximum $N^{j,c}_{BMU}$ values, is assigned at random.

Once each output SOM neuron is assigned a class label, test data can be classified using the E-DSOM. When an input data record (image) is processed through the E-DSOM, the label of its BMU in the output layer is assigned to the data record.

D. Computational Complexity

As mentioned, each hidden layer consists of multiple parallel layers where each patch is processed by a separate SOM (Procedure I), i.e., for each patch, **Algorithm I** is used to find the BMU and the BMU_{index} is stored in its feature map (Algorithm II steps 7 to 9). The Algorithm I executes in two phases. Phase 1 calculates the Euclidean distances between the input vector x and the SOM units and finds the best matching unit (BMU). Phase 2 updates the neuron weights in the BMU neighborhood. The computational complexity of each phase in the E-DSOM hidden layer can be expressed as follows:

$$O(K^2 N^2 N^2_{map}) \tag{4}$$

Where K^2 is the number of elements in a single patch, N^2 is the number of units in a SOM and N^2_{map} is the number of patches/SOMs. Both phases are highly parallelizable. Therefore, traversing the SOM for distance calculation and weight update can be reduced to $O(1)$ in the ideal case. Therefore, for a highly parallelized ideal implementation, the above computational complexity can be reduced to:

$$O(K^2 N^2_{map}) \tag{5}$$

In the combined sampling phase of a hidden layer, all the feature maps from P parallel layers are combined into one (Procedure II). Concatenating these arrays take linear computational complexity. Therefore, the computational complexity of creating the combined sampling map can be expressed as follows:

$$O(N^2_{map} P) \tag{6}$$

The P parallel layers can be executed in parallel. Therefore, increasing the number of parallel layers very little effect on the computational time is given in Eq. (5). However, it does affect Eq. (6). When the number of parallel layers (p) increase, the

TABLE III
THE DSOM ARCHITECTURE

	Hidden Layer 1			Hidden Layer 2			Output Layer
Dataset	Map Size	Patches (K)	Stride	Map Size	Patches (K)	Stride	Map Size
MNIST	4–24	10-20	2	15	6	1	8
GSA	4–24	3-7	2	14–16	3–5	1	5–8
SP-HAR	4–24	5-17	2	14–16	3–7	1	5–8

TABLE IV
THE E-DSOM ARCHITECTURE

Dataset	Hidden Layer 1			Output Layer
	Map Size	Patches (K)	Stride	Map Size
MNIST	4-24	10-20	2	8
GSA	4-24	3-7	2	5-8
SP-HAR	4-24	5-17	2	5-8

time taken to combine them into a combined sampling layer increase linearly.

When considering the space complexity of the E-DSOM model, the number of parameters that need to be stored per hidden layer can be approximated as follows:

$$|\theta| = PK^2N^2N_{map}^2 \tag{7}$$

The number of parameters that need to be stored linearly grows with the number of hidden layers. This can be used to infer the space complexity of the model. Unlike time complexity, space complexity grows with the increase of parallel layers.

IV. EXPERIMENTS AND DISCUSSION

This section discusses the experiments and results. The experimental setup is presented followed by a comparative analysis against DSOM and other state-of-the-art unsupervised algorithms.

A. Datasets

Three datasets were used for experimentation: 1) MNIST [38], 2) Gas Sensor Array Drift (GSAD) dataset [39], and 3) Smart Phone dataset for Human Activity Recognition (SP-HAR) [38] . All the datasets were normalized to zero mean and unit variance. For all datasets, balanced subsets of the data records were selected to alleviate the class imbalance problem. Further, data records in numerical datasets (GSAD and SP-HAR) were converted into 2D square images.

The MNIST dataset contains images of hand-written characters (digits from 0-9), each 28×28 pixels in size. The complete MNIST dataset contains 55000 train images and 10000 test images. In this work, a significantly smaller training set of 3000 images was used to reduce the classifier training time. The complete testing set (10000 images) was used to test the accuracy of the algorithms.

The GSAD dataset contains 13910 records collected from 16 chemical sensors from a gas delivery facility. The dataset contains data about 6 gases and the classifier's goal is to discriminate between the gasses. The dataset contains data collected for 36 months. In this work, only the first 21 months were used to avoid concept drift in data. A balanced dataset of 4500 records was selected and the train/test split was chosen as 2400/2100. The sensor data were rearranged to a 2D grid and is processed as an image. Since each data record consists of 121 dimensions, each data record was arranged to an 11×11 image.

SP-HAR consists of 10299 smartphone sensor records of 30 subjects performing six different daily living activities. A balanced dataset of 4792 records was selected and the train/test split of 3300/1492 was chosen. Similar to the GSAD dataset, data were rearranged to a 2D grid. Since the dataset contained 561 dimensions, the features were reduced to the closest square number (529) using information gain based feature selection. Then, each record was re-arranged into a 23×23 image.

B. Hyper-parameter and Model Architecture Selection

As mentioned in Section I, we hypothesize that due to the parallel architecture of E-DSOM, a shallower model compared to the DSOM can be used. This results in a reduction of serial operations, resulting in reduced training time. In order to test this, for all the tests, a DSOM with two hidden layers and an E-DSOM with only one hidden layer were implemented. In the E-DSOM hidden layer, two parallel layers were implemented.

Table III summarizes the architecture and the hyper-parameters chosen for DSOM for the three datasets. For the MNIST dataset, the set of hyper-parameters were selected based on the experiments done by Liu et al [25] and our previous work [33]. For the other datasets, the hyper-parameters were selected experimentally through cross-validation. For each dataset, different experiments were conducted by changing the map size and the patch size within the ranges shown in Table III.

Table IV presents details of the E-DSOM architecture. Different combinations of patch sizes and map sizes were tested within the presented ranges. Across parallel layers of the same model, different patch sizes were used, but the map size was kept the same. The shallower model of E-DSOM enabled the use of bigger patch sizes than the DSOM. In order to ensure a fair comparison of classification accuracies, the output layer of DSOM and E-DSOM was implemented with a SOM of the same size.

C. Experimental Results: MNIST

Classification accuracies for the MNIST dataset is presented in **Table V**. The DSOM was able to achieve the best test accuracy of 83.468% while E-DSOM was able to achieve 87.118 % (A 3.65% improvement).

Generalization capability was analyzed with noisy test data. **Table V and Fig. 4 (a)** show the performance of the two

TABLE V
CLASSIFICATION ACCURACY COMPARISON BETWEEN DSOM AND E-DSOM

Dataset	Model	Layer1				Train Acc	Test Acc	Test Accuracy for Different Noise level (%)						
		Patch Scale1	Patch Scale2	Map Size1	Map Size2			2	5	10	20	40	50	60
MNIST	DSOM	10	-	24X24	-	85.06± 1.94	83.47± 2.85	83.37± 3.04	83.14± 2.81	83.14± 2.68	82.39± 2.63	74.46± 2.72	62.00± 3.25	20.37± 1.55
	EDSOM	10	20	24X24	24X24	**88.04± 1.96**	**87.12± 2.41**	**87.12± 2.35**	**87.15± 2.14**	**86.88± 2.17**	**86.51± 1.87**	**79.91± 1.77**	**69.34± 1.98**	**23.63± 1.71**
GSAD	DSOM	3	-	24X24	-	83.35± 2.73	57.24± 8.63	49.76± 6.19	45.20± 3.18	38.08± 1.93	32.59± 1.45	27.12± 0.83	23.84± 0.45	21.88± 1.27
	EDSOM	3	9	24X24	24X24	**91.24± 1.19**	**72.73± 6.78**	**66.82± 5.06**	**61.19± 3.90**	**50.01± 4.89**	**37.86± 3.34**	**28.59± 3.64**	**24.12± 2.03**	**22.45± 2.41**
SP-HAR	DSOM	11	-	24X24	-	62.85± 1.08	57.88± 2.31	56.90± 3.28	55.60± 2.37	52.58± 2.32	44.81± 2.12	**27.17± 2.92**	19.52± 2.19	**17.78± 0.76**
	EDSOM	11	17	22X22	22X22	**67.39± 1.12**	**64.36± 0.28**	**63.22± 1.22**	**61.90± 0.83**	**58.22± 1.57**	**48.51± 2.69**	24.14± 1.46	**19.69± 0.88**	17.35± 1.01

TABLE VI
GENERALIZATION ERROR COMPARISON BETWEEN DSOM AND E-DSOM

Dataset	Model	Generalization Errror (%) for different Noise Levels								Computational Time (s)
		0	2	5	10	20	40	50	60	
MNIST	DSOM	1.59	1.69	1.92	1.92	2.67	10.60	23.06	64.69	3788
	EDSOM	**0.92**	**0.92**	**0.89**	**1.16**	**1.53**	**8.12**	**18.69**	**64.41**	**3114**
GSAD	DSOM	26.10	33.59	38.15	45.27	**50.76**	**56.23**	**59.51**	**61.46**	390
	EDSOM	**18.51**	**24.42**	**30.05**	**41.23**	53.38	62.65	67.12	68.78	**313**
SP-HAR	DSOM	4.97	5.95	7.25	10.27	**18.04**	**35.68**	**43.32**	**45.07**	3098
	EDSOM	**3.04**	**4.18**	**5.49**	**9.18**	18.88	43.25	47.70	50.04	**2602**

algorithms. There was no significant difference in classification accuracy for both models until the noise level increased beyond 20%. Despite the drop in accuracy beyond 20% noise, it was observed that E-DSOM consistently outperformed the DSOM. Further, E-DSOM showed a lower generalization error at all the noise levels **(see Table VI)**.

When **computational time** was compared **(Table VI)**, it was observed that the E-DSOM was able to reduce the training time by more than 670 seconds compared to the DSOM (17% improvement).

D. Experimental Results: GSAD

Classification accuracies for the GSAD dataset are presented in Table V. DSOM achieved 57.24% as its best classification accuracy while E-DSOM achieved 72.73% (A 15.49% improvement).

Generalization capability: Classification accuracies with noisy data are presented in Table V and Figure 4 (b). E-DSOM outperformed DSOM at all noise levels. Further, the E-DSOM showed a lower generalization error at noise levels of 0%-20% (see Table VI).

When **computational time** was considered (Table VI), it was observed that the E-DSOM was able to finish training more than 70 seconds faster than the DSOM with GSAD dataset (19% improvement).

E. Experimental Results: SP-HAR

Classification accuracies for the SP-HAR dataset are presented in Table V. DSOM was able to achieve a maximum test accuracy of 57.88% while E-DSOM was able to achieve 64.36 (6.48% improvement).

In terms of **generalization capability**, E-DSOM outperformed DSOM at all noise levels except at 40% and 60% (Table V and Figure 4(c)). Further, E-DSOM showed a lower generalization error for 0%-10% noise levels (see Table VI).

Table VI shows the **computational times** of the two models. E-DSOM was able to finish training over 490 seconds faster than the DSOM for SP-HAR dataset (around 16% improvement).

F. Overall Results Discussion

For MNIST and GSAD datasets, E-DSOM showed superior performance in classification accuracy, generalization capability and computational time.

For the SP-HAR dataset, the E-DSOM achieved superior classification performance and reduced computational time. However, the E-DSOM failed to outperform the DSOM at noise levels beyond 40%, but scores remained comparable. SP-HAR dataset contains data from smartphone sensors, which can be less precise than the industrial grade sensors in GSAD. The noisy data could be the reason for the lower classification accuracies shown by both algorithms.

The overall classification accuracy results support our hypothesis that the E-DSOM architecture with parallel layers is able to achieve better/higher accuracy with a fewer number of serial layers compared to DSOM, i.e., using less computational time.

G. Analysis: Effect of patch size and map sizes on classification accuracy

The analysis studied the effect of the two most important hyper-parameters—patch size and map size—on the classifi-

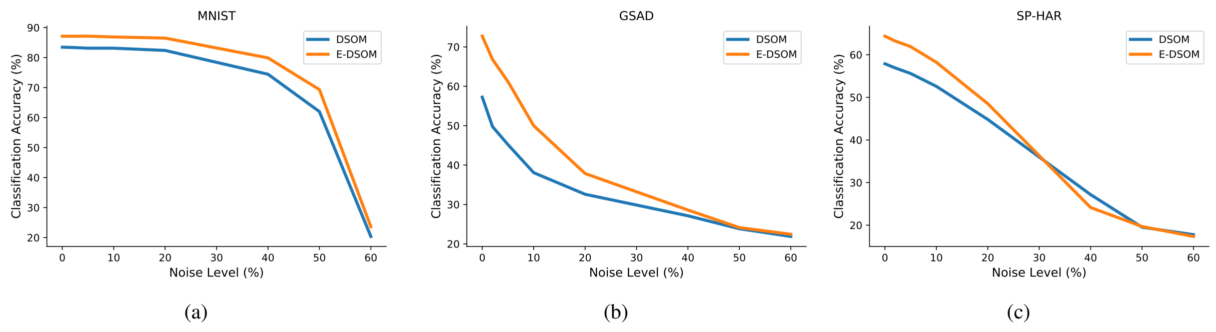

Fig. 4. Change in accuracy with different noise levels for E-DSOM and DSOM. (a) MNIST, (b) GSAD and (c) SP-HAR

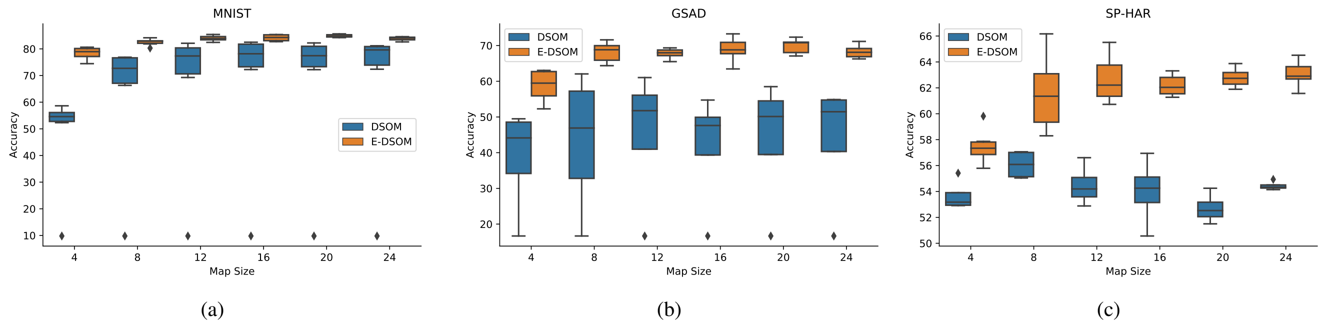

Fig. 5. Effect of patch size and map sizes on classification accuracy for E-DSOM and DSOM: (a) MNIST, (b) GSAD, (c) SP-HAR

cation accuracy. We used square maps of the size range 4–24 and different square patch sizes with each map size. For simplicity, the parameters were changed only in the first layer of both algorithms.

For MNIST and GSAD datasets, patch sizes in the range of 10–20 pixels (per dimension) and 3–7 pixels were used, respectively. The patch size was incremented in two pixels between tests. Patch sizes for SP-HAR were kept within the range of 5-17 and incremented in four pixels between tests (See Table I and II).

The results from the analysis are given in box and whisker graphs (See Figure 5(a) (c)). Each box plot relates to specific map size. The height of the box plot indicates the variability of classification accuracy for the different patch sizes, i.e., a shorter box plot indicates low variability classification accuracies and vice-versa.

Effect of Map Size: Classification accuracies were observed for different map sizes. For all tests, E-DSOM outperformed the DSOM in classification accuracy. Further, for all datasets, if the map size wasn't very low, E-DSOM's classification accuracies remained consistent across map sizes. Conversely, DSOM showed significant variations in its classification accuracies across map sizes with the exception of MNIST. Further, the smallest map size yielded the smallest classification accuracy for both models. This is expected as a small map can be inadequate to capture the feature space.

Therefore, from these datasets, it can be inferred that for the E-DSOM, as long as the map size is not too small, the classification accuracies will not change much with the map size. However, with the DSOM, in order to find the optimal map size, a thorough cross-validation process is needed.

Effect of Patch Size: As mentioned, for each map size, several patch sizes were tested. The E-DSOM consistently outperformed the DSOM despite different configurations. With the exception of SP-HAR dataset, the classification accuracies remained fairly consistent across different patch sizes with E-DSOM (shorter box plots). However, in DSOM, the results varied significantly across patch sizes for a single map size (taller box plots). In the SP-HAR dataset, the E-DSOM algorithm showed some variability for the patch sizes when the map size was 8 and 12. Therefore, it can be inferred that generally, the E-DSOM algorithm shows less dependency on the patch sizes when compared to the DSOM. This leads to an easier process of hyper-parameter selection. This could be a result of E-DSOM balancing out the effect of patch size by detecting complementary features of different resolutions in the parallel layers.

H. Comparison of E-DSOM with other unsupervised algorithms

The proposed E-DSOM architecture was compared against three other unsupervised algorithms: 1) single layer SOM, 2) stacked Autoencoder and 3) stacked Convolutional Autoencoder. A single layer SOM with an 8×8 neuron grid was implemented for the completeness purpose.

Stacked Autoencoders (AE) are deep unsupervised learning architectures [9]. AE consists of two functions, an encoder, and a decoder. Encoder learns a compressed representation of the input data and decoder reconstructs the input data using the compressed representation. AEs are widely used for dimensionality reduction [40], feature learning and data denoising [23]. In this paper, AEs was implemented with a SOM (8×8) connected to the last hidden layer. The SOM was

TABLE VII
COMPARISON OF TEST ACCURACIES OF UNSUPERVISED ALGORITHMS

Dataset	Test Accuracy (%)				
	SOM	DSOM	E-DSOM	Stacked AE	Stacked CAE
MNIST	71.26	83.47	**87.12**	84.24	81.93
GSAD	66.62	57.24	**72.73**	63.59	70.12
SP-HAR	62.80	57.88	64.36	**67.41**	66.47

trained with the encoded data, and the same classifier as E-DSOM was implemented. AEs with up to three hidden layers were tested and the best classification results are reported.

Stacked Convolutional Autoencoders (CAEs) are a variant of AEs that contains convolutional layers. CAEs are unsupervised learning algorithms, which use the building blocks—convolution layers and max-pooling layers—of supervised CNNs [40]. Similar to AE, CAE was implemented with up to three hidden layers followed by a SOM classifier. The number of filters was changed within the range 4–30. The kernel size was set to 3×3 as it resulted in the best classification accuracies. ReLU activation function was used for the non-linear transformations.

Table VII presents the test accuracy comparison between algorithms. For **MNIST**, E-DSOM achieved the best accuracy while AE came in second. Single layers SOM showed the lowest accuracy for the MNIST. For **GSAD**, E-DSOM yielded the best accuracy while CAE showed the second best accuracy. DSOM showed the lowest accuracy for the GSAD dataset. For the **SP-HAR** dataset, the AE and CAE came in first and second respectively, in terms of classification accuracy. DSOM showed the lowest accuracy for the SP-HAR dataset.

V. CONCLUSIONS AND FUTURE WORK

This paper presented a deep self-organizing map architecture (E-DSOM) for unsupervised image classification. The E-DSOM extended the originally proposed Deep Self-Organizing Maps (DSOM), in two ways: 1) the learning algorithm was modified to be completely unsupervised, 2) the architecture was modified to learn features of different resolutions in parallel. The modifications were made to improve the following: 1) classification accuracy, 2) generalization capability and 3) training time. E-DSOM was tested on three datasets and compared with DSOM. E-DSOM outperformed DSOM in terms of classification accuracy with improvements of up to 15%. Generalization capability was tested by adding noise to test data. E-DSOM outperformed DSOM at all noise levels (barring one instance with comparable results), evidencing better generalization capability. Computational time was improved by gaining the same or better classification accuracies with a shallower model. E-DSOM showed training time improvements up to 19%. Therefore, empirical evidence supports our hypothesis.

Further, E-DSOM architecture was compared to other unsupervised algorithms. E-DSOM showed comparable performance to the AE and the CAE while outperforming them on two datasets. Therefore, empirical results show that E-DSOM algorithms are competitive and a viable option for unsupervised learning.

As future work, the following avenues will be explored: 1) capability of creating low dimensional embedding of high dimensional datasets using E-DSOM; 2) capability of using E-DSOM as lightweight networks which can match with design requirements of mobile and embedded applications (MobileNets [10]), 3) incorporating negative data-mining [41] and cross-modal learning [42] to E-DSOM.

REFERENCES

[1] L. Wen, X. Li, L. Gao, and Y. Zhang. A New Convolutional Neural Network Based Data-Driven Fault Diagnosis Method. *IEEE Transactions on Industrial Electronics*, PP(99):1–1, 2017.

[2] Michal Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29*, pages 3844–3852. Curran Associates, Inc., 2016.

[3] D. Wu, X. Luo, G. Wang, M. Shang, Y. Yuan, and H. Yan. A Highly-Accurate Framework for Self-Labeled Semi-Supervised Classification in Industrial Applications. *IEEE Transactions on Industrial Informatics*, PP(99):1–1, 2017.

[4] F. Luo, Z. Dong, G. Chen, Y. Xu, K. Meng, Y. Chen, and K. Wong. Advanced Pattern Discovery-based Fuzzy Classification Method for Power System Dynamic Security Assessment. *IEEE Transactions on Industrial Informatics*, 11(2):416–426, April 2015.

[5] M. Cococcioni, B. Lazzerini, and S. L. Volpi. Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques. *IEEE Transactions on Industrial Informatics*, 9(4):2256–2263, November 2013.

[6] H. Akagi. New trends in active filters for power conditioning. *IEEE Transactions on Industry Applications*, 32(6):1312–1322, November 1996.

[7] J. R. Stack, T. G. Habetler, and R. G. Harley. Fault classification and fault signature production for rolling element bearings in electric machines. *IEEE Transactions on Industry Applications*, 40(3):735–739, May 2004.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper With Convolutions. pages 1–9, 2015.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861 [cs]*, April 2017. arXiv: 1704.04861.

[11] Friedhelm Schwenker and Edmondo Trentin. Pattern classification and clustering: A review of partially supervised learning approaches. *Pattern Recognition Letters*, 37:4–14, February 2014.

[12] J. F. Martins, V. Ferno Pires, and A. J. Pires. Unsupervised Neural-Network-Based Algorithm for an On-Line Diagnosis of Three-Phase Induction Motor Stator Fault. *IEEE Transactions on Industrial Electronics*, 54(1):259–264, February 2007.

[13] M. R. G. Meireles, P. E. M. Almeida, and M. G. Simoes. A comprehensive review for industrial applicability of artificial neural networks. *IEEE Transactions on Industrial Electronics*, 50(3):585–601, June 2003.

[14] Sanghoon Lee and M. M. Crawford. Unsupervised multistage image classification using hierarchical clustering with a bayesian similarity measure. *IEEE Transactions on Image Processing*, 14(3):312–320, March 2005.

[15] S. Lee and M. M. Crawford. Hierarchical clustering approach for unsupervised image classification of hyperspectral data. In *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages 941–944 vol.2, September 2004.

[16] Zoltan Kato, Josiane Zerubia, and Marc Berthod. Unsupervised parallel image classification using Markovian models. *Pattern Recognition*, 32(4):591–604, April 1999.

[17] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International Journal of Neural Systems*, 19(04):295–308, August 2009.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680. MIT Press, 2014.

[19] Hlne Paugam-Moisy and Sander Bohte. Computing with Spiking Neuron Networks. In Grzegorz Rozenberg, Thomas Bck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 335–376. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[20] E. Goodman and D. Ventura. Effectively using recurrently-connected spiking neural networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 3, pages 1542–1547, Montreal, QC, Canada, 2005. IEEE.

[21] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F. Wang. Generative adversarial networks: introduction and outlook. *IEEE/CAA Journal of Automatica Sinica*, 4(4):588–598, 2017.

[22] Zhifei Zhang. Generative Adversarial Networks (GANs) Past, Present, and Future. page 27.

[23] Christos Ferles, Yannis Papanikolaou, and Kevin J. Naidoo. Denoising Autoencoder Self-Organizing Map (DASOM). *Neural Networks*, 105:112–131, September 2018.

[24] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, May 2000.

[25] N. Liu, J. Wang, and Y. Gong. Deep Self-Organizing Map for visual classification. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2015.

[26] Chathurika Wickramasinghe, Kasun Amarasinghe, Daniel Marino, and Milos Manic. Deep Self-Organizing Maps for Visual Data Mining. In *11th International Conference on Human System Interaction*, Gdask, Poland, July 2018.

[27] Teuvo Kohonen. Self-Organization of Very Large Document Collections: State of the Art. In *ICANN 98*, pages 65–74. Springer, London, 1998.

[28] Thore Graepel, Matthias Burger, and Klaus Obermayer. Self-organizing maps: Generalizations and new optimization techniques. *Neurocomputing*, 21(1):173–190, November 1998.

[29] Cenk Budayan, Irem Dikmen, and M. Talat Birgonul. Comparing the performance of traditional cluster analysis, self-organizing maps and fuzzy C-means method for strategic grouping. *Expert Systems with Applications*, 36(9):11772–11781, November 2009.

[30] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, November 1998.

[31] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384, October 1996.

[32] A. Rauber, D. Merkl, and M. Dittenbach. The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, 13(6):1331–1341, November 2002.

[33] C. S. Wickramasinghe, K. Amarasinghe, and M. Manic. Parallalizable deep self-organizing maps for image classification. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, November 2017.

[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[35] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Computer Vision ECCV 2014*, pages 818–833. Springer, Cham, September 2014.

[36] R. Kumar, A. Banerjee, and B. C. Vemuri. Volterrafaces: Discriminant analysis using Volterra kernels. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 150–155, June 2009.

[37] Pengfei Zhu, Lei Zhang, Qinghua Hu, and Simon C. K. Shiu. Multi-scale Patch Based Collaborative Representation for Face Recognition with Margin Distribution Optimization. In *Computer Vision ECCV 2012*, pages 822–835. Springer, Berlin, Heidelberg, October 2012.

[38] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[39] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramn Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166-167:320–329, May 2012.

[40] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[41] Z. Ma, X. Chang, Y. Yang, N. Sebe, and A. G. Hauptmann. The Many Shades of Negativity. *IEEE Transactions on Multimedia*, 19(7):1558–1568, July 2017.

[42] Minnan Luo, Xiaojun Chang, Zhihui Li, Liqiang Nie, Alexander G. Hauptmann, and Qinghua Zheng. Simple to Complex Cross-modal Learning to Rank. *arXiv:1702.01229 [cs, stat]*, February 2017. arXiv: 1702.01229.

Chathurika S. Wickramasinghe (brahmanacsw@vcu.edu) received her B.Sc. degree in computer science from the University of Peradeniya, Sri Lanka, in 2016. She is currently working as a research assistant while reading for her doctoral degree in computer science at Virginia Commonwealth University, Richmond. Her research interests include machine learning, unsupervised learning, explainable AI, generalization and visual data mining.

Kasun Amarasinghe (amarasinghek@vcu.edu) received his B.Sc. in computer science from the University of Peradeniya, Sri Lanka, in 2011. He is currently a research assistant and a doctoral candidate at Virginia Commonwealth University. His research interests include explainable machine learning, human-machine interaction, and fuzzy systems. He is a Student Member of the IEEE and a member of the IEEE Industrial Electronics Society.

Milos Manic (misko@ieee.org) (SM06-M04-StM96) received the Dipl.Ing. and M.S. degrees in electrical engineering and computer science from the University of Ni, Ni, Serbia in 1991 and 1997 respectively, and the Ph.D. degree in computer science from the University of Idaho in 2003. Dr. Manic is a Professor with Computer Science Department and Director of VCU Cybersecurity Center at Virginia Commonwealth University. He completed over 30 research efforts in the area of data mining and machine learning applied to cybersecurity, critical infrastructure protection, energy security, and resilient intelligent control. Dr. Manic has given over 30 invited talks around the world, authored over 180 refereed articles in international journals, books, and conferences, holds several U.S. patents and has won 2018 R&D 100 Award for Autonomic Intelligent Cyber Sensor (AICS). He is an officer of IEEE Industrial Electronics Society, founding chair of IEEE IES Technical Committee on Resilience and Security in Industry, and general chair of IEEE IECON 2018, IEEE HSI 2019