Geo-AI to Aid Disaster Response by Memory-Augmented Deep Reservoir Computing

Konstantinos Demertzis[a*], Lazaros Iliadis[a], Elias Pimenidis[b]
[a]*School of Civil Engineering, Faculty of Mathematics Programming and General courses, Democritus University of Thrace, Kimmeria, Xanthi, Greece,*
[b] *Faculty of Environment and Technology, Department of Computer Science and Creative Technologies, University of the West of England, Bristol, UK*

Abstract. It is a fact that natural disasters often cause severe damage both to ecosystems and humans. Moreover, man-made disasters can have enormous moral and economic consequences for people. A typical example is the large deadly and catastrophic explosion in Beirut on 4 August 2020, which destroyed a very large area of the city. This research paper introduces a Geo-AI disaster response computer vision system, capable to map an area using material from Synthetic Aperture Radar (SAR). SAR is a unique form of radar that can penetrate the clouds and collect data day and night under any weather conditions. Specifically, the *Memory-Augmented Deep Convolutional Echo State Network* (MA/DCESN) is introduced for the first time in the literature, as an advanced Machine Vision (MAV) architecture. It uses a meta-learning technique, which is based on a memory-augmented approach. The target is the employment of Deep Reservoir Computing (DRC) for domain adaptation. The developed Deep Convolutional Echo State Network (DCESN) combines a classic Convolutional Neural Network (CNN), with a Deep Echo State Network (DESN), and analog neurons with sparse random connections. Its training is performed following the Recursive Least Square (RLS) method. In addition, the integration of external memory allows the storage of useful data from past processes, while facilitating the rapid integration of new information, without the need for retraining. The proposed DCESN implements a set of original modifications regarding training setting, memory retrieval mechanisms, addressing techniques, and ways of assigning attention weights to memory vectors. As it is experimentally shown, the whole approach produces remarkable stability, high generalization efficiency and significant classification accuracy, significantly extending the state-of-the-art Machine Vision methods.

Keywords: Geo-AI, Disaster Response, Domain Adaptation, Meta-Learning, Synthetic Aperture Radar, Echo State Network, Deep Reservoir Computing, Memory-Augmented Architecture.

1. Introduction

A disaster that leads to many casualties is a great challenge for all services involved in rescue and support. Immediate assistance by any possible means is required, to make the best possible decisions, under difficult and adverse conditions in an environment of panic and increased risk. Disaster response mechanisms should be able to collect information immediately; to support mapping the area of interest, and compare conditions before and after the disaster. Decisions need to be made about options such as, finding the most suitable location for rescue vehicles, sorting stations, and first aid kits. In addition, decisions must be made on the necessary equipment, the allocation of priorities, and the scheduling of required human resources to support rescue services. The case of the Beirut explosion is a typical example [1]. The huge explosion in the port of the Lebanese capital, came from 2,750 tons of stored ammonium nitrate. It destroyed a very large area of the city, especially the port of Beirut, while hundreds of people were trapped in the wreckage of buildings.

In cases like this, aerial observations and satellite images, could be particularly valuable in dealing with the crisis. Even when the weather does not allow traditional electro-optical sensors to get a clear picture, SAR shots offer significant help [2]. SAR can penetrate the clouds and collect high resolution data under all weather conditions, day and night. However, despite its undoubted advantages, this approach can be reduced to an inquiry tool, limited to the observation ability of humans.

This risk gives rise to the need of and demand for distancing from human intervention and the engagement of advanced Computer Vision technologies and Artificial Intelligence (AI). These will make it possible to automatically extract essential information, such as spatial-temporal area

[*]*Konstantinos Demertzis, School of Civil Engineering, Faculty of Mathematics Programming and General courses, Democritus University of Thrace, Kimmeria, Xanthi, Greece, E-mail: kdemertz@fmenr.duth.gr, Website: https://utopia.duth.gr/~kdemertz*

comparisons and object recognition, in almost real-time.

This paper introduces the M-A/DCESN, a Geo-AI disaster response computer vision system, which uses memory-augmented deep reservoir computing for domain adaptation. It aims to record, map and identify a disaster area, using materials from SAR.

The proposed system offers a meta-learning technique, which implements a reservoir computing system, using memory-augmented methods. More specifically, it employs a DCESN network which allows the storage of useful data from past processes, by integrating external storage memory. At the same time, it facilitates the rapid integration of new information, without the need for retraining the network.

The main contribution of this work is that it enhances scientific and technical knowledge about meta-learning methods, which are based on a memory-augmented approach for domain adaptation. In order to obtain acceptable classification results, these techniques require sufficient training data to be available for every particular image and tuned hyperparameters to achieve the best performance. Obtaining accurate results is challenging, particularly for near real-time applications. Therefore, past knowledge must be utilized to overcome the lack of training data in the current regime. This challenge of domain adaptation, in which the training data (source) and the test data (target) are sampled from different domains is a considerable challenge and the performance of the proposed techniques can be significantly affected by the type of problem, the nature of the data, and the type of data shift associated with the domains. Although more data can be obtained from different sources, adapting these sources to obtain acceptable results is also a challenging task. This is especially true when the different domains contain a severely imbalanced class distribution. In this study, a novel technique, based on deep neural networks, was developed and evaluated in order to solve the domain adaptation problem for remote sensing image classification in different settings.

The proposed DCESN combines a classic CNN and a DESN with analogue neurons, with sparse random connections in the input levels and in the Dynamical Reservoir (DR). Its training process uses the RLS method in the output layer. Moreover, the proposed system is assisted by a set of original modifications to the training setting, memory retrieval mechanisms, addressing techniques, and ways of assigning attention weights to memory vectors. These facilitate the learning of specialized techniques for extracting useful intermediate representations, making full use of first and second order derivatives as a pre-training method for learning parameters, without the risk of problems such as exploding or diminishing gradients. At the same time it avoids possible overfitting, while significantly reducing training time, producing improved stability, high generalization performance, and categorization accuracy.

The rest of this paper, includes the following sections: section 2 provides a detailed review of the relevant literature. Section 3 presents the methodology followed, while section 4 analyzes in detail the implementation of the proposed architecture. Section 5 describes the data and presents the results from the experiments performed. Section 6 critically discusses the method and observations made, while Section 7 summarizes the findings and presents the future objectives of the research.

2. Related Research

The success of deep learning in the field of computer vision, has been highlighted by multiple surveys about topics such as simple [3], fast [4] and 3D object detection [5]; image recognition [6] or classification [7] by novel intelligence methods [8], pixel-level classification [9] and semantic segmentation [10].
A variety of architectures have been proposed to solve complex problems and have provided a new impetus in this area. In particular, in the field of data analysis from multispectral sensors, many architectural prototypes have been developed and have delivered impressive results. Specifically, [11] proposes a hybrid approach, which combines the use of a Stacked Auto-encoder, Principle Component Analysis (PCA), and Logistic Regression in order to perform Hyperspectral Data Classification.

Tao et al. [12], are using a sparse stacked auto-encoder, to effectively represent features from unlabeled spatial data. The learned features are used as input to a SVM for hyperspectral data classification. Various 1D [13] and 2D [14] CNN architectures, aiming to encode spectral and spatial information, have been suggested in the literature. The most recent and advanced proposal, concerns 3D CNN [15] in which the third dimension refers to the time axis. This is resulting in a hyperspectral classification that follows a spatiotemporal architecture. In 3D CNN, the convolution operations are performed both spatially and spectrally, while in 2D CNNs they are performed only spatially.

Compared to 1D and 2D CNNs, 3D CNNs can better format spectral information due to the contribution of 3D convergence functions.

More sophisticated techniques inspired by dynamic architectures have raised additional expectations for even more important innovative applications in the field of spectral analysis. A typical example is our proposal [16] for a major modification that upgrades the well-known Residual Neural Network (ResNet) architecture. The network is effectively simplified, by eliminating the Vanishing Gradient Problem (VGP) which plagues other deep learning architectures. This is achieved by omitting some layers in the early training stages. The most important innovation of the proposed system concerns the use of the AdaBound algorithm that uses dynamic limits in its learning rates, achieving a smooth transition to stochastic gradient techniques. This fact treats the noisy scattered points of incorrect classification with great precision, something that other spectral classification methods cannot handle. Our research team has proposed the Model-Agnostic Meta-Ensemble Zero-shot Learning (MAME-ZsL) [17], which facilitates the learning of specialized techniques for extracting useful intermediate representations in complex deep learning architectures. This significantly reduces computational cost and training time, producing remarkable classification accuracy.

MAME-ZsL follows a heuristic, hierarchical hyperparameter search methodology. It uses the intermediate representations extracted from other possibly irrelevant images, so that it can discover the appropriate representations that can lead to correct classification of unknown samples.

Visual perception often involves sequential inference over a series of intermediate goals of growing complexity towards the final objective. Depending on a deep learning architecture, the graph can also contain extra nodes that explicitly represent tensors between operations. In such representations, operation nodes are not connected directly to each other, rather using data nodes as intermediate stops for data flow. If data nodes are not used, the produced data is associated with an output port of a corresponding operation node that produces the data.

In order to achieve a smooth formalization, the notion of intermediate concepts points to better generalization through deep supervision, when compared to standard end-to-end training. This is achieved by a strict architecture where hidden layers are supervised with an intuitive sequence of intermediate concepts, in order to incrementally regularize the learning to follow the prescribed inference sequence. Practically the intermediated representations produce superior generalization capability that addresses the scarcity of learning shape patterns from synthetic training images with complex multiple object configurations.

Domain adaptation [18], partial domain adaptation [19] and domain alignment [20] is a relatively recent forecasting technique for target domain data. Both supervised and unsupervised methods have been used, which in most of the cases try to minimize domain deviation, while neglecting essential class information. This often results in misalignment and poor generalization performance. To address this issue, the authors of [21] propose a Contrastive Adaptation Network (CAN) that explicitly models domain-level mismatch, while also calculating the difference between classes. This technique performs relatively well against similar methods, producing distinctive features, but lags far behind in terms of generalization as it is completely determined by the available data. Accordingly, the success of unsupervised sector adaptation relies heavily on the alignment of capabilities between sectors.

A common feature space may not always be a training tool and in particular an immediate alignment feature, especially when large domain gaps are observed. To solve this problem, the authors of [22] introduce a Gaussian guided Latent Alignment approach, aiming to align the latent feature distributions of two domains. The implementation delivers an innovative alignment of features between sectors, transforming the distributions of samples from two sectors into a common feature space. Despite the enhanced knowledge transfer capabilities, this method adds significant complexity to the system. Memory-Augmented Neural Networks (MANNs) have been shown to outperform other repetitive neural networks in a number of sequence [23] learning tasks [24]. However, they still have limited application in real world problems. An evaluation of MANNs applications is performed in [25].

Finally, an impressive approach is presented in [26], where a One-shot Learning approach is performed, using an augmented-memory neural network. It yields accurate predictions using only a few training samples.

3. Discussion on the Methodological approach

Recent developments in the field of information technology and especially in the techniques of high-capacity models, such as deep neural networks, allow very powerful implementations in the processing of large-scale data [27]. Nevertheless, the disclosure of critical knowledge from large-scale datasets, and in particular the correct classification of new, unknown data, combined with a parallel automatic correction of classification errors, remains a very serious challenge.

A potential solution to this problem is offered by meta-learning techniques [28] as specific "*learning to learn*" models [29]. They learn from previous learning processes, or from previous classification tasks that have been completed [30]. This is a subfield of machine learning where advanced learning algorithms are applied to data and metadata of a given problem.

In general, input patterns with and without tags come from the same boundary distribution or follow some common cluster structure. This is the case for modeling situations of real physical problems [30]. Thus, the classified data contribute to the learning process, while useful information can be extracted from the unclassified data, for the exploration of the data structure of the general set. This information can be combined with knowledge from previous learning procedures, or previous classification tasks performed.

Based on the above, meta-learning techniques can discover the underlying structure of data, allowing for fast learning of new tasks. This is achieved by using different types of knowledge, such as the properties of the learning problem, the properties of the algorithm used (e.g. performance measures) or patterns derived from data related to a previous problem. Cognitive information from unknown examples sampled from the distribution of real-world cases is used. The goal is to enhance [28] the outcome [29] of the learning process [30].
In this way it is possible to learn, select, change, or combine different learning algorithms to effectively solve a given problem.

A meta-learning system [28] should combine [29] the following requirements [30]:

1. The system must include a learning subsystem.
2. Experience should be gained from the use of knowledge extracted from metadata, related to the dataset under consideration or from previous learning tasks, completed in similar or different fields.
3. The learning bias should be selected dynamically.

Depending on the approach, there are four meta-learning prototypes as mentioned below [30]:

1. Model-based: These are techniques based on the use of retrospective networks with external or internal memory. These techniques quickly update their parameters with minimal training steps, which can be achieved through their internal architecture, or by using control from other models.
2. Memory-Augmented: Neural Networks and Meta Networks are typical model-based meta-learning techniques.
3. Metrics-based: These are techniques based on learning effective distance measurements that can generalize. The core idea of their operation is similar to that of the "*Nearest Neighbors"* whereas their goal is to learn a measurement or distance from objects. The concept of a good metric depends on the problem, as it should represent the relationship between the inputs in the space, facilitating problem solving. Convolutional Siamese Neural Network, Matching Networks, Relation Networks and Prototypical Networks are typical cases of metrics-based and meta-learning approaches.
4. Optimization-based: They are based on the optimization of the model's parameters in order to achieve fast learning. *LSTM Meta-Learners, Temporal Discreteness* and the *Reptile* algorithm are typical cases of optimization-based, meta-learning techniques.

Recurrent Neural Networks (RNN) with only internal memory and Long Short-Term Memory methods (LSTM), are not considered as meta-learning approaches. Literature suggests that memory capacity neural networks provide a meta-learning [28] approach [29] for deep neural networks [30]. However, this particular memory usage strategy that is inherent to unstructured iterative architectures, is unlikely to extend to settings where each new task requires significant amounts of new information for rapid encoding [30].

A scalable solution has some essential requirements. Information must be stored in memory in a representation that is stable, so that it can be reliably accessed when needed and addressed with data. In this way, it can selectively access relevant data. The number of parameters must not be related to the size of the memory. These two features do not occur in the original retrospective memory network architectures such as RNNs or more advanced ones such as LSTMs. In contrast, architectures such as Neural Turing Machines (NTMs) [31] and Memory Networks [32] meet the required criteria.

This research introduces the M-A/DCESN approach that uses external storage memory, which is compiled by employing the NTMs architecture. It allows memorization of useful information from past processes, while facilitating the rapid integration of new information, without the need for retraining.

4. Implementation

The NTMs are a model-based meta-learning [30] architecture and they constitute the implementation of a neural control mechanism with external storage memory. Specifically, it is an architecture that connects a neural network and an external memory storage unit. Taking a general approach to MA/DCESN in terms of its meta-learning properties [28-30], it trains in a variety of learning tasks.

It is optimized to provide a/for a better performance in generalizing tasks, including potentially unknown cases. Each task is associated with a data set *D*, containing feature vectors and class labels on the given supervised learning problem. The optimal parameters of the model are [30, 33-34]:

$$\theta^* = arg_{\theta}^{min} \mathbb{E}_{D \sim P(D)}[L_\theta(D)] \quad (1)$$

Although it seems similar to a normal learning process, each data set is still considered a sample of data.

The dataset *D* comprises two parts, a training set *S* and a testing set *B* for validation and testing [30, 33-34].

$$D = \langle S, B \rangle \quad (2)$$

D contains pairs of vectors and labels so that [30, 33-34]:

$$D = \{(x_i, y_i)\} \quad (3)$$

Each tag belongs to a known set of tags *L*.

In the case of the classifier f_θ, parameter θ extracts a probability of the class *y* render of attributes vector, $x, P_\theta\ (y|x)$.

Optimal parameters maximize the likelihood of finding true tags in multiple training batches. $B \subset D$ [30, 33-34]:

$$\theta^* = argmax_\theta \mathbb{E}_{(x,y) \in D}[P_\theta(y|x)] \quad (4)$$

$$\theta^* = argmax_\theta \mathbb{E}_{B \subset D}\left[\sum_{(x,y) \in B} P_\theta(y|x)\right] \quad (5)$$

The aim of the model is to reduce prediction error in data samples with unknown tags, considering that there is a small set of support for fast learning which works as "fine-tuning".

A modification of the model is shown in the following function, to which the symbols of the meta-learning process have been added

$$\theta^* = argmax_\theta \mathbb{E}_{L_s \subset L}\left[\mathbb{E}_{S^L \subset D, B^L \subset D}\left[\sum_{(x,y) \in B^L} P_\theta(x, y, S^L)\right]\right] \quad (6)$$

As for the model in terms of the augmented-memory technique, memory stores processed information.

It can be considered as a *N*×*M* matrix. The control mechanism is a DCESN which is responsible for performing tasks in memory.

The controller processes the input and interacts with the memory bank to generate the output, through a recurring update process. A general description of the function of the proposed NTM [31] is shown in figure 1 below.

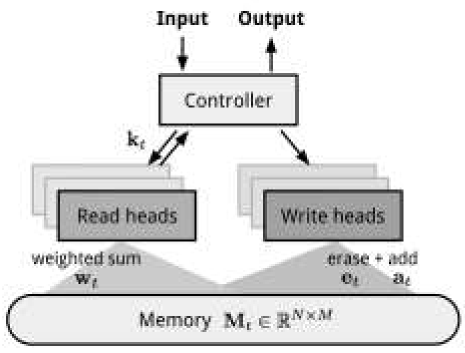

Fig. 1. Architectural modeling of the NTM [31]

When the memory is read at time *t,* an attention vector w_t of magnitude N controls how much attention should be allocated to different memory locations.

Vector r_t is the sum of the weights from the attention intensity resulting from the assignment process.

The overall calculation procedure is presented by the following equation [31, 37]:

$$r_i = \sum_{i=1}^{N} w_t(i)\, M_t(i), where \sum_{i=1}^{N} w_t(i) = 1, \forall i: 0 \le w_t(i) \le 1 \quad (7)$$

Where, $w_t(i)$ is the ith element in w_t and $M_t(i)$ is the ith element stored in memory.

In addition (inspired by forgetting gates in LSTM) the process of writing to memory in time *t* initially provides for the deletion of the old erasable vector e_t which is the content of memory in a specific location. Then new information is inserted by adding vector a_t. This procedure is described below in the corresponding deletion equation 8 and addition equation 9 [31, 37]:

$$\widehat{M}_t(i) = M_{t-1}(i)[1 - w_t(i)e_t] \quad erase \quad (8)$$

$$M_t(i) = \widehat{M}_t(i) + w_t(i)a_t \quad add \quad (9)$$

The way of the development of the attention distribution w_t depends on the addressing mechanisms, which operate on the basis of content or location.

The content-based addressing process, generates attention vectors based on the similarity between the k_t key vector (extracted by the controller from the input lines) and the memory content.

Content-based attention scores are calculated as the cosine of similarity between the content, which is then normalized with the use of the *softmax* function.

In addition, a power multiplier β_t is added to enhance or soften the focus of attention distribution.

The procedure is described in the following equation [31, 37]:

$$w_t^c(i) = softmax(\beta_t \cdot cosine[k_t, M_t(i)]) = \frac{exp\left(\beta_t \frac{k_t \cdot M_t(i)}{\|k_t\| \cdot \|M_t(i)\|}\right)}{\sum_{j=1}^{N} exp\left(\beta_t \frac{k_t \cdot M_t(i)}{\|k_t\| \cdot \|M_t(i)\|}\right)} \quad (10)$$

A step-by-step gateway is then used to mix in the last step of the time, the newly created content-based attention vector with the attention weights [31, 37]:

$$w_t^g = g_t w_t^c + (1 - g_t) w_{t-1} \quad (11)$$

On the other hand, location-based addressing gathers values at different positions in the attention vector, weighted based on a weight distribution relative to permissible integer displacements.

They are equivalent to a 1-d convolution with kernel $s_t(.)$.Finally, the attention distribution is enhanced by a gradual escalation $\gamma_t \geq 1$. The above procedures are described in the following equations 12 (circular convolution) and 13 (sharpen) [31, 37]:

$$\widehat{w}_t(i) = \sum_{j=1}^{N} w_t^g(j)\, s_t(i - j) \quad (12)$$

$$w_t(i) = \frac{\widehat{w}_t(i)^{\gamma^t}}{\sum_{j=1}^{N} \widehat{w}_t(j)^{\gamma^t}} \quad (13)$$

This work is based on the MA/DCESN architecture [8], proposing a set of modifications regarding the training setting, memory recovery mechanisms, addressing techniques and ways of assigning attention weights to memory vectors.

In particular, the main concern of the proposed system, is related to the development of a training process that uses memory capable of rapid encoding and recording information for new tasks. Moreover, any stored representation should be easily and stably accessible. Training should be performed in a way that memory can hold information for a longer time, until the appropriate labels that fit the categorization process are presented.

In each training cycle, the actual tag is presented following a step shift (x_{t+1}, y_t), so that this label (while it is part of the time step input *t*) can be part of the input in the next time step *t+1*. Following this process, the proposed MA/DCESN is motivated to memorize the information of a new data set. Memory has to hold the current input until the label appears and then the old information has to be retrieved in order for a similar prediction to be produced.

In addition to the training process, an innovative addressing mechanism is used, where the reading attention process is constructed solely on the basis of the similarity of the content.

This procedure first predicts a key vector attribute k_t in the time step *t* as a function of input *x*.

A gravity reader w_t^r of the N elements is calculated as the similarity between the cosine of the key vector and each line of the memory vector, normalized to *Softmax* [33] as follows in equation 14.

$$w_t^r(i) = softmax(cosine[k_t, M_t(i)]) \quad (14)$$

Additionally, the vector reader r_i is a sum of weighted memory files. Its mathematical description is presented in the following equation15 [31, 37]:

$$r_i = \sum_{i=1}^{N} w_t^r(i)\, M_t(i), where\ w_t^r(i) = softmax\left(\frac{k_t \cdot M_t(i)}{\|k_t\| \cdot \|M_t(i)\|}\right) \quad (15)$$

Where, M_t is a memory matrix for time stamp *t* and $M_t(i)$ is the ith line of the table.

The memory updating, for efficient retrieval and storage of information, is performed based on the Least Recently Used Access (LRUA) algorithm. This writes new content either to the least used memory location, based on the *Least Frequently Used* (LFU) algorithm, or to the most recently used memory location based on the *Most Recently Used algorithm* (MRU) [37]. Specifically, LFU is used to retain the most frequently used information.

One of the most serious weaknesses of this method is the fact that new data entering memory may be removed very soon. This may happen because they receive a very low counter, although they may be used very often after this assignment. Accordingly, the MRU algorithm first removes the most recently used memory components.

This process has proven to be very effective in cases where the older elements are considered the most useful. The motivation for its use is the fact that once an information is retrieved, it will probably not be needed immediately again [37].

The proposed MA/DCESN is developed with the employment of LRUA. Another advantage of this hybrid scheme is that all of its parameters are fully customizable.

Specifically [37]:

1. The weight w_t^u which is used at time *t* is a sum of the used read and write vectors. w_{t-1}^u is the decayed last usage weight, where γ is the decay factor.
2. The write vector is an interpolation between the previous reading weight (found in the last used position) and the previous least used weight (whose position is rarely used).
   The application of the sigmoid function on the hyperparameter α is the interpolation parameter.
3. The least used weight w^{lu} is scaled according to the usage weights w_t^u, where each dimension retains the value 1 if it is less than the *n*th element and it has the value 0 in any other case [37]:

$$w_t^u = \gamma w_{t-1}^u + w_t^r + w_t^w \quad (16)$$

$$w_t^r = softmax\left(cosine\left(k_t, M_t(i)\right)\right) \quad (17)$$

$$w_t^w = \sigma(\alpha) w_{t-1}^r + \left(1 - \sigma(\alpha)\right) w_{t-1}^{lu} \quad (18)$$

$$w_t^{lu} = 1_{w_t^u(i) \leq m(w_t^u, n)} \quad (19)$$

Where $m(w_t^u, n)$ is the nth smallest element of the weight vector, w_t^u.

Finally, each memory string is updated when the least used position indicated by w_t^{lu} is equal to zero. The update process is performed based on the following equation [37]:

$$M_t(i) = M_{t-1}(i) + w_t^w(i)k_t, \forall i \quad (20)$$

The analytical procedure presented above, is used by the proposed model to facilitate the learning process and to achieve adaptation in new situations after processing with only a few samples. At the same time it allows the rapid coding of new information by using external memory storage. In general, the proposed MA/DCESN is an NTM which consists of three main parts: The Controller, the Memory Bank and the Read/Write Heads, as presented in figure 2 in Appendix 1.

The proposed architecture has 256 positions of memory, while the range of allowed position changes is obtained by circular shifts and replacement of records, based on the LRUA algorithm. It should be noted that the above parameters were obtained by following a trial and error approach. The most important decision in the architectural design of MA/DCESN is the type of neural network used as a controller. In particular, the decision to use an iterative architecture (RNN, LSTM) or a simple FNN network is very important.

An iterative controller like LSTM has its own internal memory [38]. It also has significant computational resource requirements, adding high complexity to the model and the process is much slower.

The aim was not only to prove that the proposed MA/DCESN is capable of effectively solving the given categorization problem, but also that it is able to generalize far beyond the range of training data in a feasible time and computational resources' frame. Experiments were performed and various neural network architectures were compared. The selector finally chosen to be used is an extremely fast and highly efficient DESN.

ESN [39] is an iterative neural network with input, a sparsely connected hidden reservoir layer and a simple linear readout output. The connection weights on each ESN reservoir, as well as the input weights, are random. The reservoir weights are scaled in such a way as to ensure the Echo State Property (ESP) [40]. ESP is defined as a state in which the reservoir is an "echo" of its entire entry history, which is partly determined by its architecture.

The only distinct levels of the ESN are those of input u(n) and output y(n) which are determined by the problem. The hidden levels are grouped in a DR area and their number is indistinguishable. A percentage of the neurons in DR, are interconnected. This percentage is related to the sparsity of DR which is determined experimentally [41].

The synaptic compounds that unite the levels with each other and the DR are characterized by a value that determines the weights. In ESNs, each input neuron is connected via W^{in}_{ij} weights (i-input neuron, j-neuron to DR) to each DR neuron [41]. These weights, although normalized, are determined randomly before training and their values are final as they do not change during training. Also each DR neuron is interconnected via W_{jk} weights to any other.

The weights of these neurons, although normalized, are determined randomly before training and their values do not change. Finally, each DR neuron is connected via W^{out}_{jm} weights to the neurons of the output. These weights in the readout layer, are the only ones that are trained in order to get their final values. The basic architecture of an ESN network is described in figure 1. Where u (n) is the number of neurons in the input unit, x(n) is the number of neurons in the internal unit (which is essentially DR) and y(n) is the number of neurons in the readout layer [41].

Development of a DESN Reservoir Computing architecture [42], requires the use of multiple reservoirs. A Deep Dynamical Reservoir (DDR) area is created with the properties mentioned above [43].

The DESN architecture is characterized by a stacked hierarchy of reservoirs, where at each time step *t*, the first repeating layer is fed from the external input *u*(*t*), while each successive layer is fed from the output of the previous one into the stack [42-43].

The architectural organization of DDRs in DESN allows for general flexibility in the size of each layer

Here we consider a hierarchical tank installation with repeating layers N_L, each of which contains the same number of units N_R. Moreover we use $x^{(l)}(t) \in R^{N_R}$ to declare the status of level l at time t. By omitting the bias conditions, the first level state transition function is defined as follows [42-43]:

$$x^{(1)}(t) = \left(1 - a^{(1)}\right)x^{(1)}(t-1) + a^{(1)} \tanh\left(W_{in}u(t) + \hat{W}^{(1)}x^{(1)}(t-1)\right) \quad (20)$$

For each level higher than l >1 the equation has the following form [42-43]:

$$x^{(l)}(t) = \left(1 - a^{(l)}\right)x^{(l)}(t-1) + a^{(l)} \tanh\left(W^{l}x^{l-1}(t) + \hat{W}^{(l)}x^{(l)}(t-1)\right) \quad (21)$$

Where $W_{in} \in R^{N_R \times N_U}$ is the input weight matrix, $\hat{W}^{(l)} \in R^{N_R \times N_R}$ is the recurrent weight matrix for layer l, $W^{(l)} \in R^{N_R \times N_R}$ is the matrix containing the connection weights between layer l-1 and l, $a^{(l)}$ is the leaky parameter of layer l and $tanh$ is the Tangent Hyperbolic function [42-43].

In the DESN architecture, we must determine the number of neurons in the input unit, the size of the DDR, the depth of the architecture, the training mode and the number of nodes [42] in the readout layer [43].

4.1 Input Unit

The number of neurons at the input level is usually determined by the requirements of the problem, the individual issues related to modeling at the level of available data, and the solution sought.

The weights connecting the input level and DDR are taking random normalized values, and their population number is $(K+1)\times N$ where $(K+1)$ is the number of neurons at the entry level along with the threshold.

4.2 Deep Dynamical Reservoir

The creation of DDR, presupposes that the reservoir allows previous network states to sound even after their passage. So if the network receives an input line similar to data in which it has been trained, it will follow the appropriate activation trajectory in the reservoir. This will generate the appropriate output signal and in case the network is satisfactorily tuned, it will be able to generalize from the data with which it has been trained. The reservoir acts both as a non-linear extension of the input data, but also as a memory.

It is essentially a larger non-linear representation $x(n)$ of the input data, $u(n)$. It is also used to store data as internal memory, providing temporal context. In this spirit, RNN-like architecture is used to ensure that history is preserved. The size of the reservoir is one of the most basic parameters. Larger size means easier to find a linear combination that can produce the desired result. Due to the fact that ESNs do not have very high computational costs in many cases the size of the reservoir can receive high values. The lower limit can be calculated approximately based on the desired number of values that the network should remember. So the largest number of values to be stored should not exceed N^x, i.e. the total size of the reservoir.

Also the reservoir variable sparsity, indicates how sparse the connections between DDR neurons will be. It is a parameter that is determined during the development of the network. In many approaches the use of dilute reservoir is encouraged because it gives slightly better results. However, in relation to other parameters, sparsity does not have a high priority in the sense that it does not greatly affect the functionality of the network.

Based on the DESN architecture, DDR is defined by the W_{in} and W weight vectors, which are initialized randomly and normalized based on some parameters that can be set. The scaling used on these weights is usually the same as the one used on the weights W_{in}.

The leaking rate α is an independent parameter of reservoir neurons which translates to the speed at which the network will upgrade reservoir over time.

That is, how fast the reservoir neurons will get the ideal value. The value of this variable can be derived from the time it takes for the network input to be converted to the desired output and usually the ideal value is calculated through the experimental method.

One of the most important universal parameters of the reservoir is the spectral radius of the weights W of the DDR. This parameter expresses the maximum eigenvalue of the reservoir and sets a scale on the weights W. It essentially sets the maximum value that non-zero reservoir compounds can take. It is extremely important to maintain the ESP property, based on which the retained history should fade over a long period of time and not to depend on the original network conditions.

In cases where this parameter is set to very high values, a chaotic situation develops in the network, in which the reservoir weights change uncontrollably and the network is not trained.

4.3 Deep Architecture

Deep learning systems have a Credit Assignment Path (CAP) [33] on their depth, which describes the chain of transformations from input to output and the potentially causal links between input and output.

The CAP in the proposed DESN was performed experimentally, performing tests so that each level encodes a different range of dynamic characteristics, from the intermediate representations that are extracted [33, 41]. The main idea behind the proposed DESN design method is to stop adding new layers every time the filtration process becomes negligible. That is, when during addition of new layers, no intermediate representations are provided capable to contribute towards capturing or matching of the input data to the desired network responses of the output.

In order to determine when the filtration effect becomes negligible, a thorough study was performed. It was proved experimentally following the trial and error method that the network in this set of tests tends to converge at a certain value, as we add more than 4 levels.

Our future goal is to create a heuristic algorithm for automatically determining the depth of DESN, which will be based on a search strategy technique, suitable for automatically determining the quality of the network, based on the training dataset.

The DCESN is a hybrid of two of the most prominent forms of neural networks in modern engineering, namely a CNN [14] and a DESN architecture.

The proposed DCESN introduced in this paper incorporates a classic CNN with convolutional filters with very small receptive fields 3×3. The convolutional stride and the spatial padding were defined to be equal to 1 pixel. Max-pooling is performed over 3×3 pixel windows with stride 3. The CNN architecture includes 3 convolutional layers with 4×4, 5×5 and 4×4 convolutional filters. The number of the convolutional filters for the respective layers are 32, 64 and 128. All of the convolutional layers are employing ReLU nonlinear activation function [33]:

$$ReLU(x) = max(0, x) \text{ or } ReLU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x \geq 0 \end{cases} \quad (22)$$

However in the last layer, the *Softmax* activation function is used instead of the *Sigmoid* [33]:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \qquad j = 1, \ldots, K \quad (23)$$

This is done due to the fact that *Softmax* performs better in multi-classification problems, like the one examined here, whereas the *Sigmoid* is used in binary classification tasks.

In *Softmax*, the sum of probabilities (SUP) is equal to 1 and high values have the highest probabilities. On the contrary, in *Sigmoid* the SUP must be different than 1 and the high values have high probabilities, but not the highest ones.

In the proposed model, the *Learning Rate* was set to be equal to 0.001 and the *cross-entropy error* was used as the loss function [33].

Bootstrap Sampling was employed to enhance the efficiency of the approach [44]. The reason that this technique is used in this work is that in the specific problem of high complexity, the prediction results are multivariate. This can be attributed to the sensitivity of the correlational models to the data and to the complex relationship that describes them. An important advantage of the proposed system is the fact that it offers a stable prediction mode. The overall behavior of a multiple model is less noisy than that of a single one, while for each case, the overall risk of a particularly poor choice is reduced. It is important that the dispersion of the expected error was observed to be concentrated close to the mean error value [44].

Usually, errors of precision are probabilistic. This means that the experimenter is saying that the actual value of some parameter is probably within a specified range. For example, if the half-width of the range equals one standard deviation, then the probability is about 68% that over repeated experimentation the true mean will fall within the range; if the half-width of the range is twice the standard deviation, the probability is 95%, etc.

Thus, we can use the standard deviation estimate to characterize the error in each measurement. Another way of saying the same thing is that the observed spread of values in this example is not accounted for by the reading error. If the observed spread were more or less accounted for by the reading error, it would not be necessary to estimate the standard deviation, since the reading error would be the error in each measurement.

4.4 Readout Layer

The weights connecting each neuron from the DDR to each neuron from the output layer have a population number *N*×*L*. These weights do not get random values as long as their values are determined by the network's training.

5. Dataset and Results

SAR is a unique form of radar that can penetrate the clouds, collect data under all weather conditions, day and night. Data from SAR satellites could be particularly valuable in disaster management, especially in cases where difficult weather and clouds cover the optical capabilities of traditional electro-optical sensors. Despite their advantages, there is limited open data available to researchers to investigate the effectiveness of SAR data.

The dataset used in this research is an open-ended data set, available freely from [46] and it has been used for *SpaceNet Challenge* SN6: *Multi-Sensor All-Weather Mapping*.

The dataset uses a combination of SAR and electro-optical data sets, namely half-meter SAR images from *Capella Space* and half-meter electro-optical images from *MaxV's WorldView 2* satellite [46]. The area of interest is Europe's largest port, Rotterdam, an area with thousands of buildings, vehicles and boats of various sizes. It is the ideal point to create an effective test framework for merging SAR and electro-optical data. In particular, the training dataset contained both SAR and electro-optical images, while the test and evaluation data sets contained only SAR data.

Therefore, electro-optical data can be used to preprocess SAR data in some way, such as: coloring, domain customization, or image translation, but they cannot be used to directly map buildings. The data set was structured to mimic real-world scenarios where historical electro-optical data may be available. However, simultaneous collection of electro-optics with SAR is often not possible, due to inconsistent sensor trajectories, or bad weather conditions, that can make electro-optical data useless.

The Dataset is related to the city of Rotterdam covering an area of over 120 km^2. It comprises both high resolution synthetic aperture radar (SAR) data and electro optical (EO) images of ~48,000 buildings' footprint labels [47].

The training data comprises 450 m x 450 m tiles with associated building footprint labels of SpaceNet AOI 11 – Rotterdam (39.0 GB) and the testing data are 450m x 450m tiles of SpaceNet AOI 11 Rotterdam (16.9 GB). The data is hosted on AWS (Amazon Web Services) as a Public Dataset. It is free to download from [47].

The experiments setup process of the DCESN was performed following a supervised approach. Specifically, for each input $u(n) \in R^{Nu}$ the desired outcome is $y^{target}(n) \in R^{Ny}$. Variable *n* represents discrete time and it takes values in the closed interval [1, T] where *T* is the number of the input data vectors in the training set. The desired output $y^{target}(n)$ and the actual output *y(n)* are data vectors from the SAR dataset.

The purpose of network's training is to learn from a model with output $y(n) \in R^{Ny}$, where *y(n)* identifies as accurately as possible with $y^{target}(n)$, reducing error $E(y,y^{target})$. The ultimate target is generalization ability.

Root-Mean-Square Error (RMSE) was used as the error function (11) [33]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(P_{(ij)} - T_j\right)^2} \quad (24)$$

Where $P_{(ij)}$ is the forecasted value by program *i* for a simple assumption *j* and T_j is the target value for *j*.

It should be noted that the input level neurons are essentially inactive, as long as they do not perform any calculation. Their purpose is to transmit the network input to the DDR.

The following equation was used to update the values of the neurons in DDR [42-43]:

$$\tilde{x}(n) = tanh\left(W^{in}[1; u(n)] + Wx(n-1)\right) \quad (25)$$

Where $\tilde{x}(n) \in R^{Nx}$ defines the update values for each neuron of the DDR. Also, *Tanh* is the update function, *u(n)* is the input at temporal point *n* and 1 declares the value of the threshold (bias).

The final value of the neurons in DDR is estimated by the following equation 13 where α is the leaky integration rate $\alpha \in (0, 1]$ [42-43].

$$x(n) = (1-\alpha)x(n-1) + \alpha\tilde{x}(n) \quad (26)$$

By assigning the value α=1 in the leaking rate, we can avoid to perform leaky integration in the neurons' update, thus $\tilde{x}(n) = x(n)$ [42-43].

The weights W^{in} and *W*, which contribute to the values of x(n) are initially randomized, in order to protect our data from noise that may arise in the early stages of the process. In this way we avoid arbitrarily adjusting the *x(n)* values in training and specifically the ones that lead to an abnormal network boot state.

Upgrading the neurons to the output level based on which the neurons $y(n) \in R^{Ny}$ are defined by the internal product of the output weights $W^{out} \in R^{Ny\times(1+Nu+Nx)}$ and the vector that is developed by combining the threshold value and the vectors $u(n) \in R^{Nuy(n)}$ where $x(n) \in R^{Nx}$, is calculated by the following function 14 [42-43]:

$$y(n) = W^{out}[1; u(n); x(n)] \quad (27)$$

The update of the output neurons W^{out} which changes the weights in a way that the output *y(n)* can be as close as possible to the desired result $y^{target}(n)$, is performed by the following equation 15 [42-43]:

$$W^{out} = Y^{target}X^T(XX^T + \beta I)^{-1} \quad (28)$$

It should be mentioned that β is the Optimization Parameter used to avoid overtraining.

The proposed DCESN model an online learning algorithm was used. Based on this algorithm, the weights of the network change at any time, (at each input line of the training data).

The Recursive Least Square algorithm (RLS) was used [45]. RLS operates based on the integration of the fault history in the network upgrade calculations. In this research, RLS was used to update the weights W^{out}.

The proposed algorithm is using the *Forgetting Factor* λ (FF) which exponentially defines the importance of the error history. For example if λ=1, the error history has the same weight as the network's error at this time. If λ<1, the error history affects the network over time.

This means that the error at time *n* has a higher weight than the error at time *n*-1.

The error function in the RLS algorithm is described by the following equation 16 [45].

$$E(k) = \sum_{i=1}^{k}\lambda^{k-i}e(k)^2 \quad (29)$$

The above error function includes the parameter *e(k)* which declares the difference between the desired value y^{target} and the actual output *y* for temporal moment *k* (equation 17) [45]:

$$e(k) = y^{target}(k) - y(k) \quad (30)$$

The weight update function of the RLS algorithm changes over time, for every temporal moment *k* [45]:

$$W^{out}(k+1) = W^{out}(k) + e(k)g(k) \quad (31)$$

Where *e(k)* is defined by the above equation 18 and *g(k)* is determined by the following function 19 that determines the significance of the error history in shifting weights for *x* neurons in DDR [45]:

$$g(k) = \frac{P(k-1)x(k)}{\lambda + x(k)^T P(k-1)x(k)} \quad (32)$$

Where *P(k-1)* is determined by the following equation 20 [45]:

$$P(k) = \lambda^{-1}P(k-1) - g(k)x^T(k)\lambda^{-1}P(k-1) \quad (33)$$

This is a recursive function that allows error history to be taken into account when the weights W^{out} are updated. Also λ is the forgetting factor and x the DDR neurons.

For the case of the "*SpaceNet*" Multi-Sensor All-Weather Mapping dataset, the ranking was based on the *SpaceNet* Metric (SPAN) which is using F1-Score. It is based on the intersection over union of the footprints of two buildings, with a threshold equal to 0.5. F1-Score is calculated by taking the total True Positives (TP), False Positives (FP), and False Negatives (FN) for the total number of buildings' footprints present in the testing datasets. Specifically, the F1-Score is defined by the equation below [33]:

$$\mathrm{F1-Score} = 2\mathrm{X}\frac{\frac{\mathrm{TP}}{\mathrm{TP+FP}} \mathrm{x} \frac{\mathrm{TP}}{\mathrm{TP+FN}}}{\frac{\mathrm{TP}}{\mathrm{TP+FP}} + \frac{\mathrm{TP}}{\mathrm{TP+FN}}} \quad (34)$$

The proposed approach was compared with other corresponding Deep Learning architectures, which can be summarized as follows:

1. 1-D CNN (1DCNN): The network's architecture was designed as in [48] and it includes the input, the convolutional, the max-pooling, the fully connected, and the output layers.
2. The number of convolutional filters equals 20, the length of each filter is 11 and the pooling size has the value 3. Finally, 100 hidden units are included in the fully connected layer.
3. 2-D CNN (2DCNN): The architecture was designed using the one of [49] as prototype. It comprises of three convolutional layers equipped with 4×4, 5×5 and 4×4 convolutional filters (COF). The convolutional layers –except the last one- are followed by max-pooling layers. Moreover, the number of the COF corresponding to the convolutional layers are 32, 64 and 128, respectively.
4. Simple Convolutional/Deconvolutional Network (SCDN): This is the network comprising of simple convolutional blocks. It employs the unpooling process which is applied in [50-51].
5. Residual Convolutional/Deconvolutional Network (RCDN): This architecture uses residual blocks and a more accurate unpooling function [52].

The final parameters used in each of the 4 ESNs for the development of the DESN in the context of this proposal, were determined through a trial and error procedure and are presented in table 1 below. The trial and error method was used to deliver optimal hyperparameters for a known pattern. The goal is to reduce the prediction error in data samples with unknown tags, given that there is a small set of support for fast learning that works as fine-tuning. A step-by-step example of the process run is presented below:

1. Creation of a subset of $L_s \subset L$ tags;
2. Creation of an $S^L \subset D$ training subset and a $B^L \subset D$ prediction set. Both of these subsets include labeled data belonging to the subset L_s, $y \in L_s$, $\forall(x,y) \in S^L$, B^L;
3. The optimization process uses the B^L subset to calculate the error and update the model parameters via error propagation.

Each sample pair (S^L, B^L) is also considered as a data point. Thus, the model is trained so that it can generalize to new, unknown datasets.

Table 1. ESN parameters

Parameter	Value	Explain
Max Iterations	100	Specifies the maximum number of iterations the network required for its training.
Input Size	60	Defines the number of neurons in the input layer
Reservoir Size	210	Defines the number of DR neurons, which map the distribution of the given problem's data.
Leaking Rate	0.7	It concerns the speed with which the network upgrades the reservoir in relation to time and receives values in (0, 1].
Sparsity of Reservoir	0,4	Determines how thin the reservoir is. That is, it determines the number of synaptic connections to be present in the DR, in order to ensure a balance in the mode of operation of the network.
Spectral Radius	1.25	Basic parameter of the reservoir. It is used to set a maximum value for the weights that connect the neurons to each other.
Forgetting Factor	0.6	RLS parameter defining how less important is the error history exponentially.
Optimization Parameter	1e-8	This variable is used as a measure to avoid network overtraining and it is applied to the weight upgrade equation.
Larning Rate	0.53	It is the Learning rate of the network. An mean learning rate of 0.53 was used. It uses dynamic boundaries [0.01, 0.85] aiming to overcome the low generalization performance.

The classification performance results of the proposed approach, compared to the ones obtained by other methods are presented in table 2 below. It provides information on the results of the McNemar test of the proposed network and the other approaches examined. The McNemar statistical test was employed to evaluate the importance of classification accuracy derived from different approaches:

$$z_{12} = \frac{f_{12} - f_{21}}{\sqrt{f_{12} + f_{21}}}$$

where f_{ij} is the number of correctly classified samples in classification i, and incorrect one are in classification j. McNemar's test is based on the standardized normal test statistic, and therefore the null hypothesis, which is "no significant difference," rejected at the widely used $p = 0.05$ ($|z| > 1.96$) level of significance.

We have used hardware based on the GPU chipset, optimized for deep learning software TensorFlow.

Table 2. Classification Performance

	1DCNN	2DCNN	SCDN	RCDN	M-A/DCESN
OA	80.87	82.91	81.96	83.68	89.74
Precision	80.95	82.90	82.00	83.70	89.80
Recall	81.00	82.95	81.95	83.70	89.75
F1-Score	81.00	82.90	82.00	83.70	**89.75**
avg5ETT*	698 sec	881 sec	704 sec	751 sec	623 sec
McNemar	35.988	34.311	34.706	35.624	35.545

** average of the 5 epochs training time produced by 10 repeats of the methodology*

As can be seen from the comparative results, the proposed MA/DCESN has achieved improved results in relation to the respective competing systems.

One of the main advantages of the introduced system is its high reliability which is clearly shown by the high values of the F1-Score. This can be considered as the result of successful data processing that allows the retention of the most relevant data for the upcoming forecasts.

The proposed approach to reducing the generalization error is to use a larger model. This may require the use of regularization during training that keeps the weights of the model small. More specifically, regularization in the proposed methodology adds additional information to transform the ill-posed problem into a more stable well-posed problem. This leads the model to map the inputs to the outputs of the training dataset in such a way that the weights of the model are kept small. This weight decay approach has proven very effective in the DESN model. Regularization methods like weight decay provide an easy way to control overfitting for large neural network models [70].

The integration of external memory, makes it possible to memorize useful data from past processes, while facilitating the rapid integration of new information, without the need for retraining.

The proposed standardization offers the possibility of managing multiple intermediate representations. The hierarchical organization of reservoirs in successive layers is naturally reflected in the structure of the dynamics of the developed system.

This scaling also allows the progressive classification and exploration of input data interfaces across the levels of the hierarchical architecture, even if all levels share the same hyperparameters' values. Furthermore, the multilevel architecture of the successive reservoirs, compared to the shallow ones respectively, yielded a dynamic behavior that represents a transitional state of how the internal representations of the input signals are determined [71]. This leads to high performance even for problems that require long internal memory intervals.

Correspondingly, the hierarchical set of reservoirs is more efficient in cases where short-term network capabilities are required, than the corresponding shallow architectures, which would have to work with the same total number of iterative or recursive units in order to achieve similar results [72].

Accordingly, in terms of computational efficiency, the introduction of a multilevel construction of reservoirs in the design of a neural system, also results in a reduction in the number of non-zero repetitive connections on many-core architectures [73]. This implies low complexity and time savings, which is required to perform specialized tasks as presented in Table 2. Also, segmentation maps can be produced as soon as at least a single satellite image acquisition has been successfully and subsequently improved, once additional imagery becomes available. In this way, we are able to reduce the amount of time needed to generate satellite imagery-based disaster damage maps, enabling first responders and local authorities to make swift and well-informed decisions in responding to disasters.

6. Conclusion

This paper proposes a novel *Geo-AI* disaster response computer vision system that uses meta-learning memory-augmented Deep reservoir computing for domain adaptation. The purpose is to map a disaster area [53-60] using SAR radar material, which can penetrate the clouds and collect data day and night and in all weather conditions.

The reliability of the proposed system was tested in the recognition of scenes from remote sensing images in the *SpaceNet Multi-Sensor All-Weather Mapping* dataset. This fact proves its capacity to be used in higher level Geospatial Data Analysis processes, such as multidisciplinary classification, recognition, and monitoring of specific patterns. It can also be used in the fusion of SAR and multi sensors' data for disaster response [74-76].

7. Further work

The proposals for evolution and future development of MA/DCESN, focus on the development of reservoirs with Spiking neurons. These types of neurons require minimum training time, they do not require delicate manipulations in determining their operating parameters, and they can determine the appropriate output weights for the most efficient solution of a problem.

Also, it would be important to study the expansion of this system by implementing more complex architectures in an environment of parallel and distributed systems that share the same memory.

Moreover, we aim to enhance the research by newer and more powerful supervised machine learning/classification algorithms such as Enhanced Probabilistic Neural Network [77], Neural Dynamic Classification algorithm [78], Dynamic Ensemble Learning Algorithm [79], and Finite Element Machine for fast learning [80].

Finally, a future extension would be the development of a network with methods of self-improvement and automatic redefining of its parameters. This would result in a heuristic algorithm for determining the depth of DCESN, which will be based on an ensemble [81] search strategy [82], suitable for the automatic determination of the networks' quality based on the training set.

References

[1] https://en.wikipedia.org/wiki/2020_Beirut_explosion

[2] https://earthdata.nasa.gov/learn/what-is-sar

[3] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," arXiv:1911.09070 [cs, eess], Jul. 2020, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1911.09070.

[4] Z. Cai, Q. Fan, R. S Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. ECCV, pages 354–370, 2016

[5] J. Lehner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Nessler, and S. Hochreiter, "Patch Refinement -- Localized 3D Object Detection," arXiv:1910.04093 [cs], Oct. 2019, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1910.04093.

[6] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou, "Fixing the train-test resolution discrepancy: FixEfficientNet," arXiv:2003.08237 [cs], Apr. 2020, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/2003.08237.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Conference on Computer Vision and Pattern Recognition, June 2016.

[8] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. F.-Fei, A. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In International Conference on Computer Vision, September 2018

[9] A. Tao, K. Sapra, and B. Catanzaro, "Hierarchical Multi-Scale Attention for Semantic Segmentation," arXiv:2005.10821 [cs], May 2020, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/2005.10821.

[10] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu. Ccnet: Criss-cross attention for semantic segmentation. arXiv:1811.11721, 2018.

[11] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, Deep learning-based classification of hyperspectral data, IEEE Journal of Applied Earth Observations & Remote Sensing, vol 7, no 6, pp. 2094–2107, 2014

[12] C. Tao, H. Pan, Y. Li, Z. Zou, Unsupervised spectral-spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification, Geosc. Rem. Sensing, vol 8, no 6, pp 2381–2392,

[13] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," IEEE Geoscience and Remote Sensing Letters

[14] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, Deep supervised learning for hyperspectral data classification through convolutional neural networks, Geosc. & Rem. Sensing, 2015

[15] Y. Chen, H. Jiang, C. Li, X. Jia, P. Ghamisi, Deep feature extraction and classification of hyperspectral images based on CNN,Transactions on Geosc. & Rem. Sensing, vol 54, no 10, pp 6232–6251, 2016.

[16] K. Demertzis, L. Iliadis, E. Pimenidis (2020) Large-Scale Geospatial Data Analysis: Geographic Object-Based Scene Classification in Remote Sensing Images by GIS and Deep Residual Learning. In: Iliadis L., Angelov P., Jayne C., Pimenidis E. (eds) Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society, vol 2. Springer, Cham. https://doi.org/10.1007/978-3-030-48791-1_21

[17] K. Demertzis, L. Iliadis, GeoAI: A Model-Agnostic Meta-Ensemble Zero-Shot Learning Method for Hyperspectral Image Analysis and Classification. Algorithms 2020, 13, 61.

[18] J. Liang, D. Hu, and J. Feng, "Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation," arXiv:2002.08546 [cs], Aug. 2020, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/2002.08546.

[19] Z., Cao, K., You, M., Long, J., Wang, and Q, Yang. Learning to transfer examples for partial domain adaptation CVPR 19.

[20] F. M., Cariucci, L., Porzi, B., Caputo, E., Ricci, and S. R. Bulo, Autodial: Automatic domain alignment layers. In ICCV, 2017

[21] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive Adaptation Network for Unsupervised Domain Adaptation," arXiv:1901.00976 [cs], Apr. 2019, Accessed: Aug. 30, 2020.

[22] J. Wang, J. Chen, J. Lin, L. Sigal, and C. W. de Silva, "Discriminative Feature Alignment: Improving Transferability of Unsupervised Domain Adaptation by Gaussian-guided Latent Alignment," arXiv:2006.12770 [cs], Aug. 2020, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/2006.12770.

[23] A., Graves, G. Wayne, and I. Danihelka. 2014. Neural turing machines. arXiv preprint arXiv:1410.5401

[24] A., Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. G.-Barwiska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, and J. Agapiou. 2016. Hybrid computing using a neural network with dynamic external memory. Nature, 538(7626):471

[25] M. Collier and J. Beel, "Memory-Augmented Neural Networks for Machine Translation," arXiv:1909.08314 [cs, stat], Sep. 2019, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1909.08314.

[26] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot Learning with Memory-Augmented Neural Networks," arXiv:1605.06065 [cs], May 2016, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1605.06065.

[27] J. Dai et al., "BigDL: A Distributed Deep Learning Framework for Big Data," arXiv:1804.05839 [cs], Nov. 2019, doi: 10.1145/1122445.1122456.

[28] I. Khan, X. Zhang, M. Rehman and R. Ali, "A Literature Survey and Empirical Study of Meta-Learning for Classifier Selection," in IEEE Access, vol. 8, pp. 10262-10281, 2020, doi: 10.1109/ACCESS.2020.2964726.

[29] S.; Hochreiter, A.S.; Younger, P.R. Conwell, Learning to Learn Using Gradient Descent. In Proceedings of the ICANN'01 International Conference, Vienna, Austria, 21–25 August 2001; pp. 87–94.

[30] C.; Lemke, M.; Budka, B. Gabrys, Metalearning: A survey of trends and technologies. Artif. Intell. Rev. 2013, 44, 117–130, doi:10.1007/s10462-013-9406.

[31] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," arXiv:1410.5401 [cs], Dec. 2014, Accessed: Aug. 30, 2020.

[32] J. Weston, S. Chopra, and A. Bordes. 2015. Memory networks. In Proceedings of the 2014 International Conference on Learning Representations (ICLR). arXiv preprint arXiv:1410.3916.

[33] J. Schmidhuber, 2015 "Deep learning in neural networks: An overview". Neural Networks. 61: 85–117.

[34] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. IEEE Press, 2001.

[35] C.; Finn, P.; Abbeel, S. Levine, A.Nichol and J.Achiam and John Schulman (2018). On First-Order Meta-Learning Algorithms. arXiv 2017, arXiv:1803.02999.

[36] C.; Finn, S. Levine, Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. arXiv 2017, arXiv:1710.11622.

[37] A., Santoro, S. Bartunov, M. M. Botvinick, D. Wierstra and T. P. Lillicrap. "Meta-Learning with Memory-Augmented Neural Networks." ICML (2016).

[38] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," arXiv:1402.1128 [cs, stat], Feb. 2014, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1402.1128.

[39] E. A., Antonelo, E. Camponogara, and B. Foss, (2017). Echo State Networks for data-driven downhole pressure estimation in gas-lift oil wells. Neural Networks 85, 106-117.

[40] M. Buehner and P. Young (2006) A tighter bound for the echo state property. IEEE Transactions on Neural Networks, 17(3):820-824

[41] M. Lukoševičius (2012) A Practical Guide to Applying Echo State Networks. In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686

[42] C. Gallicchio and A. Micheli, "Deep Echo State Network (DeepESN): A Brief Survey," arXiv:1712.04323 [cs, stat], Feb. 2019, Accessed: Aug. 30, 2020. [Online]. Available: http://arxiv.org/abs/1712.04323.

[43] C. Gallicchio, A. Micheli, and L. Pedrelli, "Design of deep echo state networks," Neural Networks, vol. 108, pp. 33–47, Dec. 2018, doi: 10.1016/j.neunet.2018.08.002.

[44] R. Polikar, (2006). "Ensemble based systems in decision making". IEEE Circuits and Systems Magazine. 6 (3): 21–45. doi:10.1109/MCAS.2006.1688199

[45] X., Qing J., Xu F., Guo A., Feng W., Nin H. Tao (2007) An Adaptive Recursive Least Square Algorithm for Feed Forward Neural Network and Its Application. In: Huang DS., Heutte L., Loog M. (eds) Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. ICIC 2007. Lecture Notes in Computer Science, vol 4682. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74205-0_35

[46] https://spacenet.ai/

[47] aws s3 ls s3://spacenet-dataset/spacenet/SN6_buildings/

[48] W.; Hu, Y.; Huang, L.; Wei, F.; Zhang, H. Li, Deep convolutional neural networks for hyperspectral image classification. J. Sens. 2015, 2015, 258619.

[49] Y.; Chen, H.; Jiang, C.; Li, X.; Jia, P. Ghamisi, Deep feature extraction and classification of hyperspectral images based on CNN. Trans. Geosci. Remote Sens. 2016, 54, 6232–6251.

[50] A.; Dosovitskiy, J.T.; Springenberg, T. Brox, Learning to Generate Chairs, Tables and Cars with Convolutional Networks. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 692–705.

[51] A.; Dosovitskiy, P.; Fischer, J.T.; Springenberg, M.; Riedmiller, T. Brox, Discriminative unsupervised feature learning with exemplar convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell. 2016, 38, 1734–1747.

[52] L.; Mou, P.; Ghamisi, X.X. Zhu, Unsupervised Spectral–Spatial Feature Learning via Deep Residual Conv–Deconv Network for Hyperspectral Image Classification. IEEE Trans. Geosci. Remote Sens. 2018, 56, 391–406.

[53] C.,Zhang, W., Yao, Y., Yang, R., Huang, and A. Mostafavi, (2020), Semi-automated Social Media Analytics for Sensing Societal Impacts due to Community Disruptions during Disasters, Computer-Aided Civil and Infrastructure Engineering, 35:12, 1331-1348 (18 pages)

[54] X. Pan, T.Y. Yang, (2020), Post-disaster imaged-based damage detection and repair cost estimation of reinforced concrete buildings using dual convolutional neural networks, Computer-Aided Civil and Infrastructure Engineering, 35:5, 495-510.

[55] A., Lenjani, C.M., Yeum, S.J., Dyke, I. Bilionis, (2020), Automated Building Image Extraction from 360-degree Panoramas for Post-Disaster Evaluation, Computer-Aided Civil and Infrastructure Engineering, 35:3, 241-257.

[56] A., Nejat, R.J., Javid, S., Ghosh, and S. Moradi, (2020), A Spatially Explicit Model of Post-Disaster Housing Recovery, Computer-Aided Civil and Infrastructure Engineering, 35:2, 150-161.

[57] C. Fan, and A. Mostafavi, (2019), A Graph-based Method for Social Sensing of Infrastructure Disruptions in Disasters, Computer-Aided Civil and Infrastructure Engineering, 34:12, 1055-1070.

[58] M., Xu, M., Ouyang, Z., Mao, X. Xu, (2019), "Improving Repair Sequence Scheduling Methods for Post-disaster Critical Infrastructure Systems," Computer-Aided Civil and Infrastructure Engineering, 34:6, 506-522.

[59] X. Liang, (2019), "Image-Based Post-Disaster Inspection of Reinforced Concrete Bridge Systems Using Deep Learning with Bayesian Optimization," Computer-Aided Civil and Infrastructure Engineering, 34:5, 415-430.

[60] Z., Wang, M., Guo, H., Hu, and J. Gong, (2019), "Optimization of Temporary Debris Management Site Selection and Site Service Regions for Enhancing Post-Disaster Debris Removal Operations," Computer-Aided Civil and Infrastructure Engineering, 34:3, 230-347.

[61] F.J., Vera-Olmos, E., Pardo, H., Melero, and N., Malpica, DeepEye: Deep Convolutional Network for Pupil Detection in Real Environments, Integrated Computer-Aided Engineering, 26:1, 2019, pp. 85-95.

[62] T., Yang, C., Cappelle, Y., Ruichek, and M., El Bagdouri, Multi-object Tracking with Discriminant Correlation Filter Based Deep Learning Tracker, Integrated Computer-Aided Engineering, 26:3, 2019, pp. 273-284.

[63] P., Lara-Benıteza, M., Carranza-Garcıa, J., Garcıa-Gutierreza J.C., Riquelme, Asynchronous dual-pipeline deep learning framework for online data stream classification, Integrated Computer-Aided Engineering, 27:2, 2020, 101-119.

[64] R.A., Sørensen, M., Nielsen, and H., Karstoft, Routing in congested baggage handling systems using deep reinforcement learning, Integrated Computer-Aided Engineering, 27:2, 2020, 139-152.

[65] K., Thurnhofer-Hemsi, E., López-Rubio, N., Roé-Vellvé, and M.A., Molina-Cabello, Multiobjective optimization of deep neural networks with combinations of Lp-norm cost functions for 3D medical image super-resolution, Integrated Computer-Aided Engineering, 27:3, 2020, 233-251.

[66] J., García-González, J.M., Ortiz-de-Lazcano-Lobato, Luque-R.M., Baena, and E., López-Rubio, Background subtraction by probabilistic modeling of patch features learned by deep autoencoders, Integrated Computer-Aided Engineering, 27:3, 2020, 253-265.

[67] S., Hamreras, B., Boucheham, M.A., Molina-Cabello, R., Benıtez-Rochel Lopez-Rubio, Content-based image retrieval by ensembles of deep learning object classifiers, Integrated Computer-Aided Engineering, 27:3, 2020, 317-331.

[68] D., Simoes, N., Lau, and L.P., Reis, Exploring Communication Protocols and Centralized Critics in Multi-Agent Deep Learning, Integrated Computer-Aided Engineering, 27:4, 2020.

[69] J., Benito-Picazo, E., Domínguez, E.J., Palomo, and E., López-Rubio, Deep learning-based video surveillance system managed by low cost hardware and panoramic cameras, Integrated Computer-Aided Engineering, 27:4, 2020.

[70] I. Bougoudis, K. Demertzis, and L. Iliadis, 'Fast and Low Cost Prediction of Extreme Air Pollution Values with Hybrid Unsupervised Learning'. 1 Jan. 2016 : 115 – 127.

[71] S. Hamreras, 'Content Based Image Retrieval by Ensembles of Deep Learning Object Classifiers'. 1 Jan. 2020 : 317 – 331.

[72] S. Colreavy-Donnelly, 'Shallow Buried Improvised Explosive Device Detection via Convolutional Neural Networks'. 1 Jan. 2020 : 403 – 416.

[73] P., Emerson C., de L., D. Pereira, and T., Gianluca. 'A Multiobjective Metaheuristic Approach for Morphological Filters on Many-core Architectures'. 1 Jan. 2019 : 383 – 397.

[74] H. Luo, and S.G. Paal, (2019), " A locally-weighted machine learning model for generalized prediction of drift capacity in seismic vulnerability assessments," Computer-Aided Civil and Infrastructure Engineering, 34:11.

[75] M.H. Rafiei, and H., Adeli, "NEEWS: A Novel Earthquake Early Warning System Using Neural Dynamic Classification and Neural Dynamic Optimization Model," Soil Dynamics and Earthquake Engineering, Vol. 100, 2017, pp. 417-427.

[76] M.H. Rafiei, and H. Adeli, (2018), "A Novel Machine Learning Model for Construction Cost Estimation Taking Into account Economic Variables and Indices," Journal of Construction Engineering and Management, 144:12, 2018, 04018106 (9 pages).

[77] M. Ahmadlou, H., Adeli, "Enhanced Probabilistic Neural Network with Local Decision Circles: A Robust Classifier," Integrated Computer-Aided Engineering, 17:3, 2010, pp. 197-210.

[78] M.H. Rafiei, and H. Adeli, "A New Neural Dynamic Classification Algorithm," IEEE Transactions on Neural Networks and Learning Systems, 28:12, 2017, 3074-3083 (10.1109/TNNLS.2017.2682102).

[79] K.M., Rokibul Alam, N., Siddique, H., Adeli, "A Dynamic Ensemble Learning Algorithm for Neural Networks" Neural Computing with Applications, 32:10, 2020, 6393-6404.

[80] D.R., Pereira, M.A., Piteri, A.N., Souza, J., Papa, and H. Adeli, (2020) "FEMa: A Finite Element Machine for Fast Learning," Neural Computing and Applications, 32:10, 6393-6404 (https://doi.org/10.1007/s00521-019-04146-4).

[81] Breiman, L. Bagging Predictors. Machine Learning 24, 123–140 (1996). https://doi.org/10.1023/A:1018054314350

[82] T. Schoormann, D. Behrens, M. Fellmann and R. Knackstedt, "<> Design Principles for Supporting Rigorous Search Strategies in Literature Reviews," 2018 IEEE 20th Conference on Business Informatics (CBI), Vienna, Austria, 2018, pp. 99-108, doi: 10.1109/CBI.2018.00020.

Appendix 1

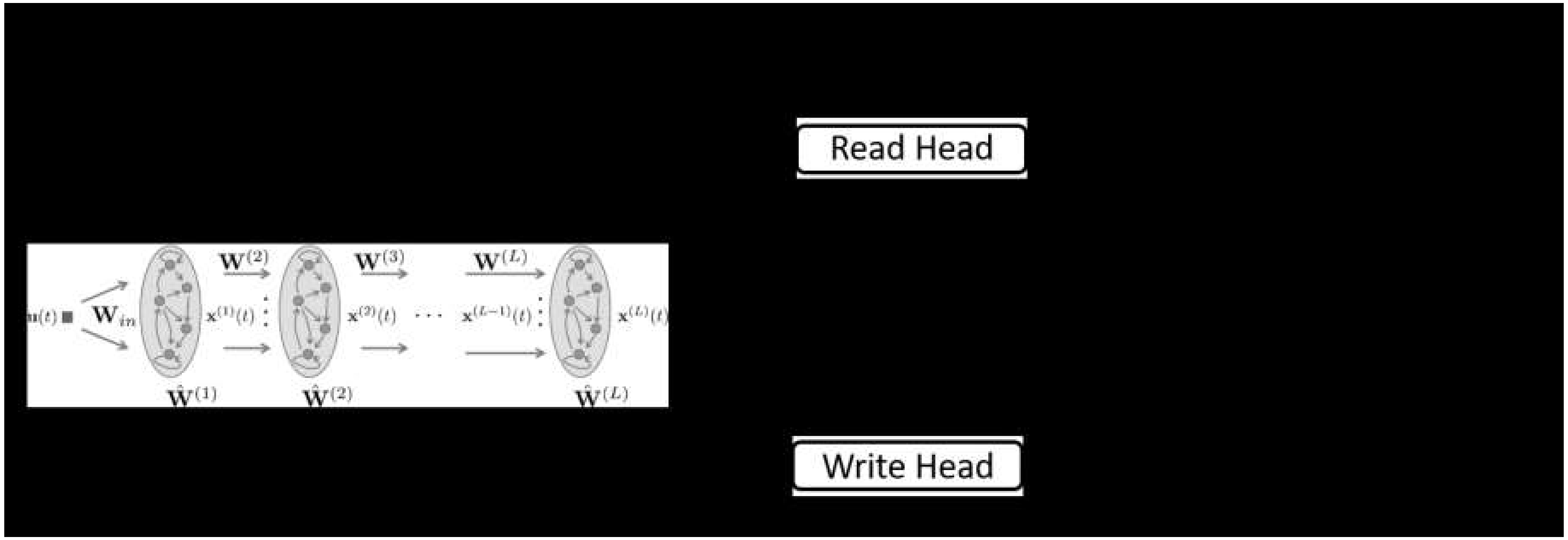

Fig. 2. Architecture of the proposed M-A/DCESN