applied sciences

MDPI

Article

Dynamic Depth Learning in Stacked AutoEncoders

Sarah Alfayez [1,2,*], Ouiem Bchir [1] and Mohamed Maher Ben Ismail [1]

[1] Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia; obchir@ksu.edu.sa (O.B.); mbenismail@ksu.edu.sa (M.M.B.I.)

[2] Department of Information and Computer Science, College of Computer Science and Engineering, University of Hail, Hail 55476, Saudi Arabia

* Correspondence: s.alfayez@uoh.edu.sa

Abstract: The effectiveness of deep learning models depends on their architecture and topology. Thus, it is essential to determine the optimal depth of the network. In this paper, we propose a novel approach to learn the optimal depth of a stacked AutoEncoder, called Dynamic Depth for Stacked AutoEncoders (DDSAE). DDSAE learns in an unsupervised manner the depth of a stacked AutoEncoder while training the network model. Specifically, we propose a novel objective function, aside from the AutoEncoder's loss function to optimize the network depth: The optimization of the objective function determines the layers' relevance weights. Additionally, we propose an algorithm that iteratively prunes the irrelevant layers based on the learned relevance weights. The performance of DDSAE was assessed using benchmark and real datasets.

Keywords: dynamic depth learning; stacked AutoEncoder; unsupervised learning; layer pruning

check for updates

Citation: Alfayez, S.; Bchir, O.; Ben Ismail, M.M. Dynamic Depth Learning in Stacked AutoEncoders. *Appl. Sci.* **2023**, *13*, 10994. https://doi.org/10.3390/app131910994

Academic Editors: Dongpo Xu, Huisheng Zhang and Jie Yang

Received: 9 September 2023
Revised: 2 October 2023
Accepted: 3 October 2023
Published: 5 October 2023

1. Introduction

Findings in the natural sciences along with recent technological advances have promoted the mimicking of the human nervous system/network. In particular, this has led to the flourishing of machine learning (ML) as a subfield of the artificial intelligence domain [1]. Specifically, ML has gained the attention of researchers who aim to design and introduce intelligent solutions for computing applications that span a very large spectrum of domains. Mainly, such exceptional interest is motivated by the availability of a huge amount of data. Moreover, drastic improvements in computing capabilities and resources have further boosted researchers' efforts to present promising ML techniques. Consequently, ML has been deployed in various domains and it has found application in various research fields such as speech and handwriting recognition [2,3], natural language processing [4], computer vision [5], image processing [6], banking and credit card fraud detection [7], etc.

The advancements in artificial neural network (ANN)-related research laid the cornerstone of deep learning (DL) [8]. In fact, the term "deep" refers to the concept of having the data transformation deployed through numerous layers [1]. Another important characteristic of deep learning is automatic feature extraction which alleviates the problem of feature handcrafting and engineering [8]. Given the success that it has achieved recently in various domains, DL has become a universal learning approach for many research communities [8]. Actually, it has proved to be effective in many fields. In particular, it yields outstanding results in image processing, computer vision, natural language processing, object detection, face recognition, and handwriting and speech recognition [1]. Furthermore, there are successful commercial applications using DL that have been established by information technology giants such as Google [9], Apple [10], Facebook [11], and Amazon [12], who use deep learning techniques in their applications.

Deep learning has been adapted for both supervised and unsupervised learning applications. AutoEncoders (AE) have been introduced as a main DL architecture designed to address unsupervised learning problems [1]. AE is a neural network composed of an input layer, a hidden layer, and an output layer. Through its hidden layer, the network learns the code which allows reproduction of the input. Such AEs can be stacked by associating successive hidden layers. This results in a deep learning topology referred to as the stacked AutoEncoder [13].

AutoEncoders are typically used for embedding learning and low-dimensional representation [13]. In addition to the dimensionality reduction and compression applications, AEs have been used for anomaly detection, image denoising, image generation, and feature extraction [13]. They have been also employed in several fields such as bioinformatics [14], cyber-security [15], recommender systems [16], etc. Furthermore, they have been integrated into other complex supervised deep learning paradigms as encoding and decoding modules [8].

Despite researchers' achievements in designing deep-learning-based solutions for various applications, some challenges still face existing DL architectures. In particular, they suffer from the interpretability limitation. Actually, DL approaches are considered black boxes that are unable to provide a logical justification of the obtained results. Another issue consists in the large number of hyperparameters to be selected and fine-tuned during the training phase. In particular, the number of hidden layers, and the number of nodes at each layer which represent the network depth and the network width, respectively, affect the model performance. In fact, overly deep and/or wide networks yield problems such as vanishing gradient [8], long training time [13], and overfitting [13]. Therefore, setting the appropriate depth and width of the network is perceived as a cornerstone in the design of an effective deep learning topology. However, determining the optimal depth and width of the network is not straightforward. These two hyperparameters are usually set empirically [17], or using some search strategies [18]. Moreover, they rely on the supervised information to compare the performance of the different settings/topologies. Thus, when the supervision information is not available, the problem is even more challenging. As such, AutoEncoders which learn, in an unsupervised way, abstract and complex feature representation of the input through the combination of the successive layers [13] face the challenging problem of determining the optimal topology in an unsupervised way. In fact, learning the topology of a stacked AE plays an essential role in its performance as it controls the level of abstraction and complexity of the learned feature map. Moreover, overly deep stacked AE are prone to the vanishing gradient problem and overfitting. Furthermore, setting the appropriate depth empirically by repeating the experiment requires the availability of labeled data, and thus can be applied for unsupervised data mapping using AutoEncoders.

The main contribution of this paper is to automatically learn the optimal number of hidden layers of a stacked AutoEncoder. Therefore, a novel cost function will be designed and optimized apart from the AutoEncoder loss function. The optimization of the proposed cost function is intended to learn relevance weights for each hidden layer. Based on the learned weights, the irrelevant hidden layers will be pruned. This yields the optimal stacked AutoEncoder depth. Moreover, pruning irrelevant layers is an acclaimed technique in reducing overfitting [19]. Hence, it can increase the overall model learning stability [19]. More specifically, the hidden layers' relevance weights will be learned simultaneously, allowing the interpretation of the role of each hidden layer. In other words, the proposed approach is intended to learn the optimum AE depth and thus alleviate the typical issues faced by overly deep stacked AutoEncoders. As such, this research has the following objectives:

(i) Learn the stacked AE depth while training the model in an unsupervised way;
(ii) Design a novel objective function to learn the layers' relevance weights;
(iii) Design an algorithm to prune irrelevant layers.

2. AutoEncoder

An AutoEncoder [1] is an unsupervised learning neural network mainly used for dimensionality reduction and compression [8]. The purpose of an AutoEncoder is to learn a representation of the input data. Furthermore, AEs are well known for data denoising and fusion [8]. The architecture of an AutoEncoder is described in Figure 1. It consists of an input layer, a hidden layer, and an output layer. More specifically, an AutoEncoder comprises two modules. The first one encodes the input while the second one decodes the obtained code to reconstruct the input. The encoder and decoder functions are characterized as lossy, where the output of the AE will be a degraded representation of the input [13].

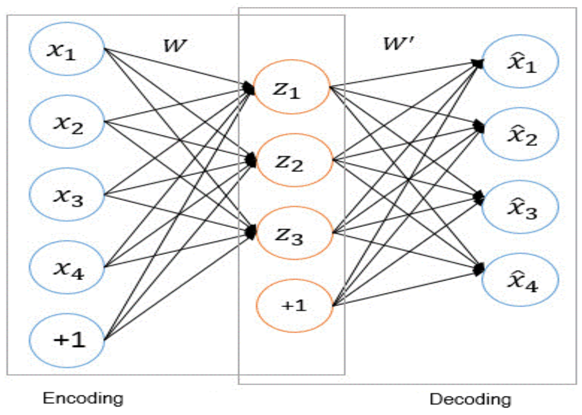


Figure 1. AutoEncoder architecture [8].

As shown in Figure 1, using the input x, the encoder generates $z_1 = \sigma_1(Wx + b)$ where σ_1 is an activation function such as sigmoid [20] or ReLU [20], W is a weight matrix, and b is a bias [8]. Alternatively, the decoder generates the construction of x, $\hat{x} = \sigma_2(W'z + b')$ where W' is a weight matrix and b' is a bias [8]. If the activation function is linear, then a similar dimensionality reduction as in principal component analysis (PCA) is achieved [21]. However, the AutoEncoder can acquire a more powerful nonlinear generalization of the PCA with nonlinear activation functions [21].

Training AutoEncoders is similar to training feedforward neural networks via back-propagation and gradient descent [21]. Basically, the network is trained in order to minimize the reconstruction error $L(x,\ \hat{x})$ which measures the differences between the original input x and its corresponding reconstruction $\hat{x}$ as defined in Equation (1) [8].

$$L(x,\hat{x}) = \|x - \hat{x}\|^2 = \left\|x - \sigma_2\left(W'(\sigma_1(Wx + b)) + b'\right)\right\|^2 \quad (1)$$

The cost function in Equation (1) may also integrate a regularization term in order to regulate the sensitivity of the AutoEncoder to the input [13]. In fact, while the function defined in Equation (1), representing the reconstruction error, is the main term for all AE loss functions, other regularization terms may be added to it according to the type of AE. Actually, an AutoEncoder has to be sensitive enough to the input to accurately build its reconstruction, but not so sensitive that the model does not simply memorize or overfit the training data.

There are various types of AutoEncoders that differ in terms of data representation, regularization technique, and architecture [13]. Namely, there are under-complete [21], over-complete [21], sparse [22], stacked [13], contractive [21], and variational [21] AutoEncoders.

Stacked AutoEncoder Topology

A stacked AutoEncoder, also referred to as a deep AutoEncoder, consists of stacking n hidden layers [1]. This allows learning more efficient code without increasing the number of nodes per layer which yields a reduction in the number of required training data, and thus reducing the training computation [21]. In fact, the performance of a stacked AutoEncoders in data compression surpasses its linear or shallower counterparts [13]. It is trained by an unsupervised greedy layer-wise learning algorithm. More specifically, the first layer is trained using the input data to produce the first feature map. This learned feature is then used as input to learn the second feature map. Thereby, the feature vector learned at the previous layer is used as an input to learn the next hidden layer consecutively. The process is repeated until all hidden layers are trained [23]. Once a feature map is learned, the respective decoder layer is discarded at [23]. Subsequently, the hidden layers are stacked to form the deep architecture of the stacked AutoEncoder [13]. Thenceforward, a backpropagation algorithm is used to adjust the weights by minimizing the error. The unsupervised training of the stacked AutoEncoder may be followed by a supervised fine-tuning algorithm [23].

The network topology is defined using the depth and width shown in the AutoEncoder architecture. Specifically, the depth represents the number of intermediate layers stacked in the encoder. One should note that this depth also equals the number of intermediate layers in the decoder. On the other hand, the width of a given layer acts for the number of nodes in this particular layer. Figure 2 illustrates the topology of a stacked AutoEncoder. As can be seen, WD_1 represents the width of the first hidden/intermediate layer of the encoder. Similarly, WD_2 and WD_3 are the width of the second and third hidden layers of the encoder, respectively. The decoder module hidden layers have the same width but in the reverse order.

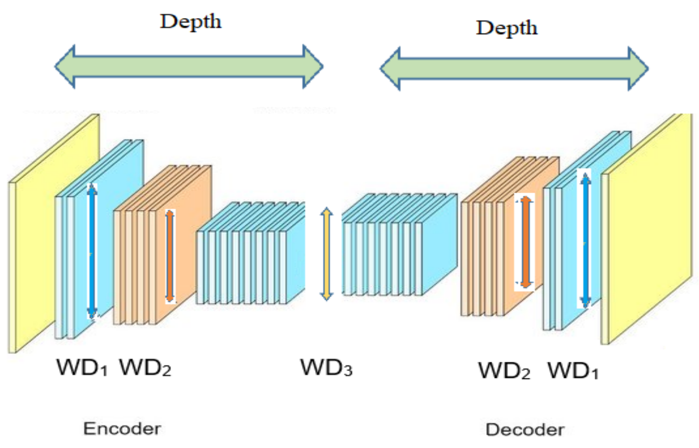


Figure 2. Stacked AutoEncoder topology.

The encoder and decoder layers of a stacked AutoEncoder can be fully connected or convolutional layers. The latter are referred to as convolutional AutoEncoders [13]. Moreover, the structure of the encoder and decoder module can consist of a sparse AE or a recurrent neural network (RNN) which is known as a sequence-to-sequence AutoEncoder [13]. Deep AutoEncoders are applied to image classification [24], regression [13], and geochemical anomaly detection [25].

3. Related Works

A key challenge in the design of a deep neural network (DNN) is the selection of the model's depth and width. Hence, there are no commonly accepted methods for configuring the network's structure. However, over the past years little research has been undertaken to find the optimal depth and width in an unsupervised manner. To the best of our knowledge, no prior research has been carried out to address the automatic learning of AutoEncoders' depth in an unsupervised fashion except the evolutionary and genetic algorithms presented in [26,27].

Recently, the problem of learning the width and the depth of deep AEs has been addressed. As such, the topology of the AE is learned while training the model through dynamic expansion and\or pruning of neurons and/or the layers [18,26–31]. These approaches are referred to as dynamic or evolving AE architecture in the literature. They bifurcate into two main categories, namely, the genetic and evolutionary approaches, and the discriminative and generative ones. While AEs are unsupervised deep learning approaches, discriminative approaches append a SoftMax layer to exploit the classification performance of the AE-generated feature map in the learning process. Thus, discriminative approaches are supervised contrary to generative genetic, and evolutionary approaches which are unsupervised. Furthermore, only genetic and evolutionary approaches learn both the width and the depth, whereas generative and discriminative approaches learn the width on a single-layer AE. In the following, we describe these approaches.

3.1. Genetic and Evolutionary Approaches

Genetic and evolutionary algorithms are metaheuristic techniques derived from the natural selection evolution mechanism. They are employed to solve optimization and search problems through the use of genetic operators such as mutation, crossover, and selection. They are defined using a genetic representation and an objective function called fitness. Specifically, starting from random solutions, a set of possible solutions referred to as individuals or candidates evolve to a better one following an iterative process. For this purpose, each solution is represented by chromosomes or genotypes, usually encoded as binary string, on which genetic operators are performed to achieve a better fitness.

3.1.1. Supervised Width Learning Based on Adding and Pruning Nodes

The proposed approach in [28] addresses the problem of learning the AE width through the use of genetic techniques. As such, it considers both adding and pruning nodes during the learning process. First, a set of AEs having different numbers of hidden units is randomly generated. Then, each one of them is separately trained by minimizing the reconstruction error which is considered the fitness value. Based on the best reached fitness, pairs of AEs are selected. Subsequently, a crossover operation is performed to produce two child AEs via an exchange of randomly chosen hidden units and their corresponding connections. Then, the mutation operation is exploited to randomly add or prune nodes from an AE. Specifically, a third AE is used to randomly select a node and add it to the child AE. Alternatively, node pruning is achieved by removing the node and its connections. Each generated child is then trained. Finally, EvoAE terminates when the last three generations fail to improve their validation error against the best obtained validation error in previous generations. The final AE architecture is defined by the best generated population according to the validation error.

3.1.2. Unsupervised Restricted Depth and Width Learning for a Multi-Layer AE

The work in [26] uses evolutionary search strategies to learn the width and the depth of a variational AutoEncoder (VAE). Nevertheless, it limits the depth to five layers and the width to a set of values that range from 50 to 1000 in steps of 50. In order to increment the depth or the width, a mutation operation is performed on selected parent genotypes based on a fitness value defined as the inverse of the reconstruction error. More specifically, the mutation operation produces an offspring by either adding a new layer or modifying the width of a randomly selected existing layer. The latter modification takes place when the limit of appending new layers is reached.

3.1.3. Unsupervised Width and Depth Learning Using a Chromosome of Fixed Length

In [27], a genetic model is adopted to learn AE width and depth. Through crossover and mutation operations on a fixed size chromosome, it produces new individuals. More specifically, the chromosome is composed of fourteen genes organized in five groups. The first group includes one gene characterizing the type of the AE. Similarly, the second group is defined by one gene estimating the number of layers (depth). Alternatively, three genes, constituting the third group, are dedicated to the number of units per layer, whereas the six genes of the fourth group define the activation functions. Finally, the last group contains one gene related to the loss function.

3.1.4. Supervised Width and Depth Learning Using a Chromosome of Variable Length

Another approach based on genetic optimization is proposed in [18]. It aims to learn the optimal depth and width of a VAE. It exploits a chromosome of variable length. The latter is constituted of four parts defining the architecture of VAEs; namely, the h-block, the μ-block, and the α-block are related to the encoder, and the t-block is associated with the decoder. The algorithm starts with a randomly generated population which is evaluated by VAE's loss function. Then, crossover and mutation operations are applied on this population. More specifically, the offspring obtained by crossover will undergo mutation operations including addition, deletion, and modification, in such a way that the addition and deletion operations act on the depth of the model, whereas, the modification operates on the width of the model.

3.2. Discriminative and Generative Approaches

Discriminative models are produced by supervised machine learning techniques which learn the class of the instances, or the border between the different categories. They predict new instances by calculating probabilities and employing maximum likelihood. Alternatively, generative models are statistical models that generate new data instances using unsupervised machine learning techniques. They are essentially based on the Bayes theory to calculate probabilities, estimate the model of the data, and cluster the instances.

3.2.1. Supervised Width Learning of Single-Layer AE based on Adding and Merging Nodes

The work in [29] learns the width of a single-layer AE by adding and merging nodes. This is achieved through a generative and discriminative hybrid training process. More specifically, it selects training instances whose construction error is greater than a predefined construction threshold. When the number of these instances reaches a pre-defined limit, a new node is added. Only the newly added nodes and their parameters are then trained by optimizing a hybrid objective function. Precisely, the generative term is the construction error of the selected set, while the discriminative term is the average classification error over the same set. The discriminative part is using the ground truth of the data. This makes it a supervised approach. Alternatively, the merging process allocates pairs of nodes with minimal distance in terms of their corresponding weights. Once merged, they are trained using the hybrid objective function.

3.2.2. Supervised Width Learning of a Single-Layer AE Based on Adding and Pruning Nodes

The work in [30] learns the width for a single-layer denoising AutoEncoder (DAE). It adopts a prequential test-then-train approach rather than the traditional train–test process. Starting from an initial one-layer DAE topology, the algorithm iteratively carries out two training phases. These consist of a generative (unsupervised pre-training) and a discriminative (supervised) phase. More specifically, before training the model, the initial model is tested. For this purpose, a SoftMax layer is appended on top of the DAE to measure the classification performance of the model. This requires the use of labeled data which makes the proposed approach supervised. After the discriminative phase, the generative phase takes place. It starts by calculating the network significance (NS) parameter which is an estimation of the network bias and variance with respect to the tested model. Actually, a high variance of the NS value indicates that the model suffers from overfitting, whereas a high bias of the NS value indicates that the model suffers from underfitting. Consequently, if the NS bias is greater than the minimum NS bias, a new hidden node is appended. Alternatively, a node is pruned if the NS variance is greater than the minimum variance. Furthermore, in order to decide on the node to be pruned, an evaluation of the contribution of each hidden node is estimated through the computation of the hidden node significance (HS) index. The learning process continues by altering the training and testing phases.

3.2.3. Unsupervised Width Learning on a Stacked AE based on Pruning Node

The work in [31] proposes to learn the layer's width of a stacked fully connected AE. The width is learned by pruning insignificant neurons. For this purpose, the reconstruction error for each node is calculated. If the reconstruction error of any neuron is 20 times larger than the minimum reconstruction error, the input data of that unit are banned from engaging in the training process. Therefore, all units subsequent to the large reconstruction error units are pruned. Table 1 depicts a brief summary of the report works in the literature that address the width or/and depth learning of an AE.

Table 1. Summary of related works.

Reference	Type of AE	Technique	Supervision	Width Learning	Depth Learning	Hidden Layer's Number
[28]	AE	Genetic	Supervised	Pruning and adding	NA	Single
[26]	Variational AE	Evolutionary	Unsupervised	Adding	Adding	Multiple (Restricted)
[27]	Stacked AE	Genetic	Unsupervised	Chromosome of fixed length	Chromosome of fixed length	Multiple
[18]	Convolutional Variational AE	Genetic	Supervised	Chromosome of variable length	Chromosome of variable length	Multiple
[29]	Denoising AE	Discriminative and generative	Supervised	Adding and merging	NA	Single
[30]	Denoising AE	Discriminative and generative	Supervised	Pruning and adding	NA	Single
[31]	Stacked Sparse Denoising AE	Generative	Unsupervised	Pruning	NA	Multiple

3.3. Discussion

A closer look at the literature relevant to dynamic model selection for stacked Auto-Encoders reveals several research gaps and shortcomings. In particular, the works in [18,27,28] which exploited evolutionary and genetic algorithms to determine the optimal architecture inherited the high computational cost of those optimization algorithms [32]. Those approaches exhibit a scalability issue especially when dealing with large networks [33]. For instance, in [32], in order to use a genetic algorithm to learn the stacked AutoEncoder depth and width in an unsupervised manner, the authors constrained the number of layers and nodes to not exceed a specific threshold [26]. The generative and discriminative approaches introduced in [29,30] to learn the topology of AEs are supervised. In other words, they relied on labeled training instances to build the stacked AutoEncoder model. The work in [31] is the only study that investigated learning stacked AE topology in an unsupervised manner. However, their approach learns the width of the network only by pruning irrelevant nodes.

This confirms that no existing work has tackled the challenge of learning the stacked AutoEncoder depth in an unsupervised fashion. Moreover, none of the state-of-the-art studies formulated this problem as an objective function-based optimization.

4. Dynamic Depth for Stacked AutoEncoders

In this paper, we propose a new approach, Dynamic Depth for Stacked AutoEncoders (DDSAE), to dynamically learn the depth of a stacked AutoEncoder. In particular, we intend to learn dynamically the relevance of each layer, l. This is done through an iterative update of the layer relevance weights at the end of each batch b. The cornerstone of the approach is the formulation of a cost function that is apart from the loss function of the network. Similar to spectral dimensional reduction approaches (PCA, MDS, Isomap, etc.) [34], the devised cost function is based on faithful preservation of the inner product between the input instance and the mapped features at each layer. More specifically, computing the difference between the inner products of the input instances and the mapped features allows preserving the moment and the argument of pairwise input instances, and thus conserves the structure of the data. Figure 3 illustrates the proposed approach during the training process. The latter is composed of three main steps that are performed iteratively. In fact, a batch b is first trained to learn the mapping of each layer l. Second, the relevance weights with respect to each layer are learned. Finally, layer pruning is performed. In the following, these training steps are described in detail.

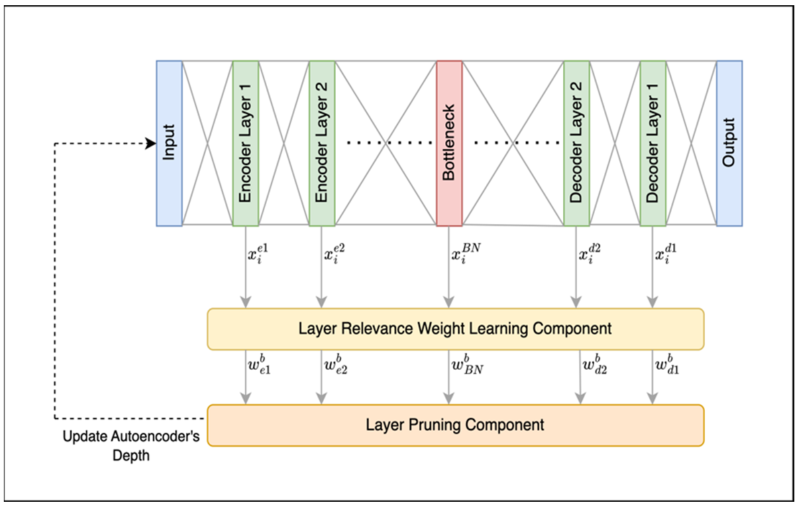


Figure 3. DDSAE training process illustration.

4.1. Layer Relevance Weight Learning Component

Learning the relevance of each layer, l, at the end of each trained batch, b, $RL(l,\ b)$ is done through the optimization of a new cost function J independent from the loss function of the AE. As such, J measures the conservation of the inner product between the input, $x(i)$, and its mapping of layer l at the end of b, $x'(i,l,b)$. It is defined as in (2).

$$J = \sum_{l=1}^{NL(b)} RL(l,b)^q \sum_{b}^{NB(b)} \sum_{i\in b} \sum_{j\in b} \left(x(i).x(j) - x'(l,i,b).x'(l,j,b)\right)^2 \quad (2)$$

subject to:

$$\sum_{l=1}^{NL(b)} RL(l,b) = 1 \quad (3)$$

In Equations (2) and (3), $NL(b)$ and $NB(b)$ are the number of layers and the number of batches trained so far at the end of batch, b, respectively. In addition, $x(i).x(j)$ is the inner product between the two inputs x_i and x_j, and $x'(l,i,b).x'(l,j,b)$ is the inner product between the mapped vectors of layer l at the end of b. Moreover, q is an exponent that controls the fuzziness of the relevance weights $RL(l,\ b)$, which is usually set to 2. The distance $(x(i).x(j) - x'(l,i,b).x'(l,j,b))^2$ measures the difference in terms of pairwise distance and angle between the input and mapped vectors. Minimizing the cost function J with respect to $RL(l,b)$ aims at learning the optimum relevance layer weights that allow preserving the moment and the argument of the input instances. Nevertheless, these weights are different from the AE model weights. In fact, they are learned from a cost function different from the loss function of the network and intend to segregate irrelevant layers.

In order to find the optimum relevance layer weights $RL(l,b)$ that minimize the cost function in Equation (2), we derive Equation (2) with respect to $RL(l,b)$ subject to

Equation (3). Then, we set the obtained derivative to zero. This gives the update equation of $RL(l,b)$ as expressed in Equation (4).

$$RL(l,b) = \frac{\frac{1}{Q(l,b)}^{\frac{1}{q-1}}}{\sum_{v=1}^{NL(b)} \frac{1}{Q(l,v)}^{\frac{1}{q-1}}} \tag{4}$$

where

$$Q(l,b) = \sum_{b}^{NB(b)} \sum_{i \in b} \sum_{j \in b} \left(x(i).x(j) - x'(l,i,b).x'(l,j,b)\right)^2 \tag{5}$$

These relevance layer weights give an interpretation of the contribution of each layer. In other words, they give insight on the importance of each layer. They are not necessarily increasing from left to right. In this regard, they will be used to prune irrelevant layers. Algorithm 1 portrays the steps of learning the relevance weights for each layer using the DDSAE approach.

4.2. Layer Pruning

After updating the AE model weights at the end of each batch, the layers' relevance weights are computed. Accordingly, non-relevant layers are pruned. Due to the hierarchical structure of the deep AE, the pruning is performed at the end of the network. More precisely, the algorithm retains the layers exhibiting encoder layers' relevance weights larger than a percentage of maximum weight threshold (W_{max}). Then, it starts from the last layer in the encoder if the layer's relevance weight is smaller than a specified threshold. As such, the layer is pruned along with its corresponding decoder layer. The process continues iteratively until the considered layer relevance weight is larger than the W_{TH}. One should note that, theoretically, in case of a data with easily separable classes, the proposed approach may result in a network with one single hidden layer. This would prove that the pruned layers are irrelevant. Algorithm 2 describes the steps of the layer pruning component.

Algorithm 1 The Layer Relevance Weight Learning Algorithm

```
Input: Instances {x_i} to be mapped
       Mapping of the instances at layer l
Output: AutoEncoder Layers' relevance weight {w_i^b}
1. Compute Q(l, b) for each layer using (5)
2. Compute the relevance weight for each layer using (4)
```

The key steps of the DDSAE algorithm are presented in Algorithm 3 and Figure 4 shows the proposed approach flowchart.

Algorithm 2 The Layer Pruning Algorithm

```
Input: AutoEncoder Layers' relevance weight {$w_i^b$} of current batch b
       Weight threshold ratio $\alpha$
Output: Trained $AE(W)$
        The learned depth
1. Separate encoder layers and decoder layers
2. Select from encoder layers the layer with the maximum relevance weight ($W_{max}$)
3. Compute $W_{TH} = \alpha * W_{max}$
REPEAT
    4. Start from last layer in the encoder If $w_i^b < W_{TH}$
    5. Prune encoder layer and corresponding decoder layer
UNTIL the $w_i^b > W_{TH}$
```

Algorithm 3 The DDSAE Algorithm

```
Input: Instances $\{x_i\}$ to be mapped
       Initialized stacked $AE(W)$ with initial depth and initial layers' width
Output: Trained $AE(W)$
        Layers' relevance weights $\{RL(l,b)\}$
        The learned depth
REPEAT
       1. Train a batch $b$
       2. Compute the layers' relevance weights $RL(l,b)$ using algorithm 2
       3. Prune irrelevant layers using algorithm 3
UNTIL the weights of the $AE(W)$ model converge
```

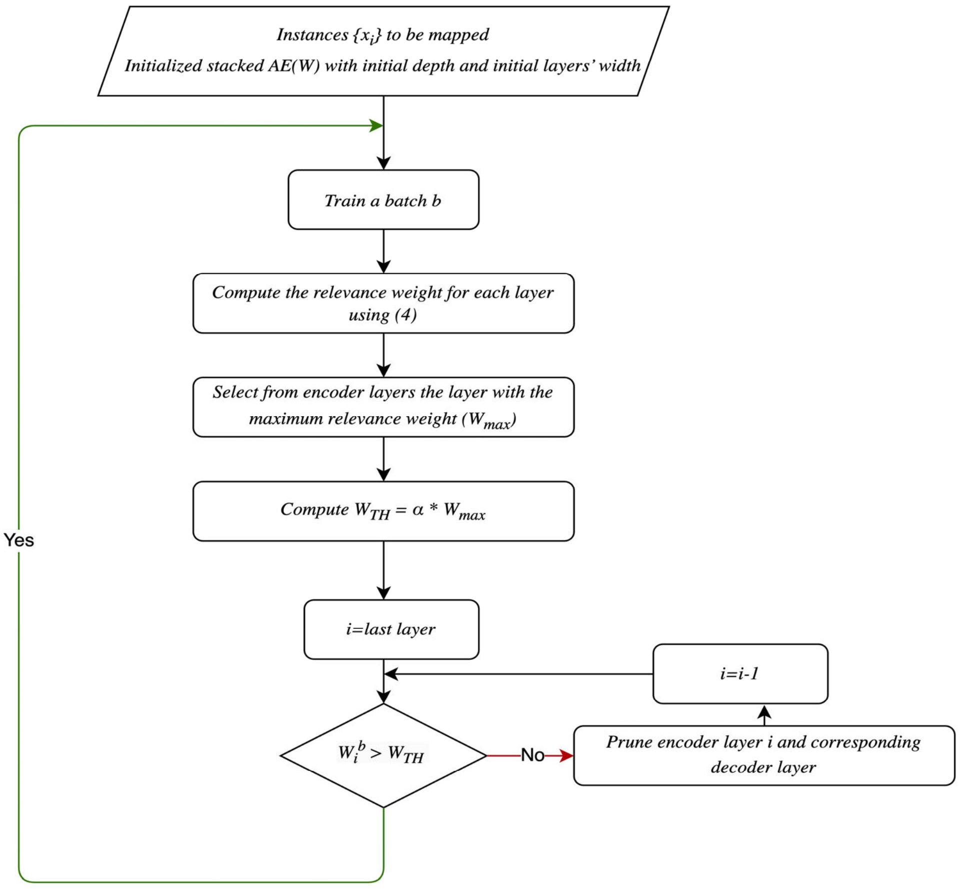

Instances {x_i} to be mapped
Initialized stacked AE(W) with initial depth and initial layers' width
Train a batch b
Compute the relevance weight for each layer using (4)
Select from encoder layers the layer with the maximum relevance weight (W_max)
Compute W_TH = α * W_max
i=last layer
i=i-1
W_i^b > W_TH
No
Prune encoder layer i and corresponding decoder layer
Yes

Figure 4. Flowchart of the proposed approach.

5. Experiments

The conducted experiments investigate the ability of the proposed approach, DDSAE, to dynamically set the depth of a stacked AutoEncoder. Specifically, experiments were conducted on two benchmark datasets and two real datasets. Moreover, the performance of DDSAE was assessed by feeding the AE learned features to a classifier, and then comparing the classification results using accuracy, precision, recall, and F1 score performance measures.

5.1. Dataset Description

MNIST—MNIST benchmark dataset [35] includes a training set of 60K examples and testing set of 10K examples. Each data instance is a black and white 28 $\times$ 28 image presenting a single handwritten digit from 0 to 9. This dataset is a balanced dataset.

CIFAR-10—CIFAR-10 benchmark dataset [36] includes 50K training images and 10K testing images. The images represent one of ten natural objects (i.e., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). Each instance is an RGB image of 32 × 32 pixels. This dataset is a balanced dataset.

Parkinson—Parkinson's disease classification dataset [37,38] consists of 756 instances, each including 754 attributes. The data collected in this dataset are from 188 patients diagnosed with Parkinson's disease and 64 healthy individuals. The attributes include various speech features extracted from the record of the phonation of vowel /a/. These features include the gender, the Pitch Period Entropy (PPE), the Detrended Fluctuation Analysis (DFA), the Recurrence Period Density Entropy (RPDE), the number of pulses (numPulses), the number of period pulses (numPeriodPulses), the mean of period pulses (meanPeriodPulses), the standard deviation of period pulses (stdDevPeriodPulses), etc. This dataset is an imbalanced dataset.

Breast Cancer—Breast Cancer Coimbra dataset [39,40] includes 116 instances with 10 attributes. The data are collected from 64 patients diagnosed with breast cancer and 52 healthy individuals. All attributes are quantitative predictors gathered in a routine blood draw analysis. The features include age, Body Mass Index (BMI), Glucose, Insulin, Homeostatic Model Assessment (HOMA), Leptin, Adiponectin, Resistin, Monocyte Chemoattractant Protein-1 (MCP.1), and classification. This dataset is an imbalanced dataset.

Table 2 reports the details of the considered datasets.

Table 2. Details of the datasets.

Dataset	Number of Samples	Features	Classes
MNIST	60,000	28 × 28 images (i.e., 784 pixels)	10 classes representing handwritten digits from 0 to 9
CIFAR-10	50,000	32 × 32 images (i.e., 1024 pixels)	10 classes representing one of ten natural objects (i.e., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)
Parkinson's	756	754 including (id, gender, PPE, DFA, RPDE, numPulses numPeriodPulses, meanPeriodPulses, stdDevPeriodPulses, etc.)	2 classes (healthy, diagnosed with Parkinson's) including 188 patients diagnosed with Parkinson's and 64 healthy patients
Breast Cancer	116	10 including (age, BMI, Glucose, Insulin, HOMA, Leptin, Adiponectin, Resistin, MCP.1, classification)	2 classes (healthy, diagnosed with breast cancer) including 64 patients diagnosed with breast cancer and 52 healthy patients

5.2. Experiment Setting

Experiments are conducted on Google Colab Pro+. In addition, TensorFlow and Keras libraries are used to implement the neural network. During training, the depth for the AutoEncoder is initially set to 20 layers for the encoder and 20 layers for the decoder. Furthermore, the width is set to 1000 nodes per layer; the weight threshold ratio is set to 0.75.

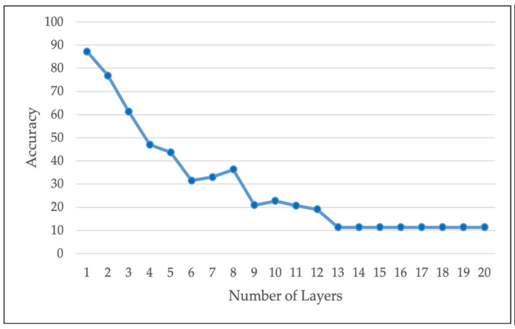
Each experiment was repeated 10 times and the best results are reported. Furthermore, the model hyperparameters were tuned and the ones giving the best results are considered. In particular, deep learning hyperparameters such as the learning rate, the batch size, and the loss function have to be set. In fact, setting suitable hyperparameters is one of the limitations of deep-learning-based experiments. They require tuning in order to ensure an optimal solution. Table 3 reports the experimental setting with respect to each dataset that was obtained.

Table 3. Parameters setting.

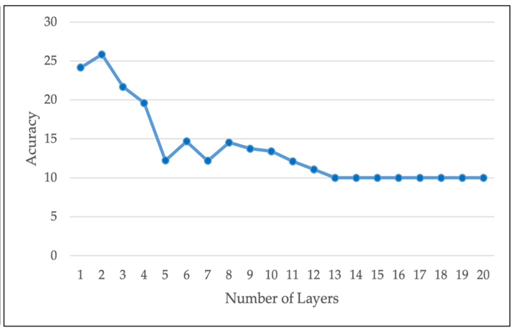
Parameter	MNIST	CIFAR	Parkinson	Breast Cancer
Optimizer	Adam	Adam	Adam	Adam
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy	Mean Squared Error	Mean Squared Error
Learning rate	0.0001	0.0001	0.000001	0.000001
Batch size	256	256	190	30

5.3. Experiment 1: Manual Depth Investigation

The first experiment manually investigates the performance of the SAE with different depth values in order to show the effect of the SAE depth on the performance and thus prove the significance of learning the depth of SAEs. In other words, the performance of the SAE is evaluated with different depths. In fact, the depth of the model is tuned from one AE architecture to twenty AE architectures. To demonstrate the impact of the network depth on the classification performance, the experiment is conducted on the two-benchmark datasets MNIST [35] and CIFAR-10 [36]. After training the SAE, a SoftMax layer is appended on the encoder part as a classifier to perform the classification task. As the two benchmark datasets are balanced, the accuracy is considered. The latter with respect to different network depths is illustrated in Figure 5.



(**a**)



(**b**)

Figure 5. Accuracy achieved when manually tuning the depth on (**a**) MNIST dataset; and (**b**) CIFAR-10 dataset.

As can be observed in Figure 5a, the best accuracy achieved is 87.20% with the one-AE model. Specifically, the classification accuracy on MNIST [35] showed that the one-AE architecture achieved a better classification accuracy compared with the 20-AE architecture. In fact, the obtained results showed that as the depth grows, the classification accuracy decreases. It is also observed that the performance of the model with depths from 1 to 4 attained better accuracy than models with deeper depths. In addition, the accuracy drastically decreases and plateaus for models with depth 13 to 20. This is a counterintuitive result showing that increasing the number of layers can decrease the accuracy.

The performance on the CIFAR-10 [36] dataset is illustrated in Figure 5b. As can be seen, the model with two-AE architecture attained an accuracy of 25.84%. The achieved results confirm the findings on the MNIST dataset, which shows that as the depth increases, the classification accuracy does not necessarily increase. Moreover, the network with two-AE architecture performs better than other SAE depths.

This steep fall in accuracy, shown in Figure 5, is related to the considered data. As can be seen, the curve displayed in Figure 5a is different from the one displayed in Figure 5b which depicts the accuracy achieved when manually tuning the depth on MNIST [35] dataset and CIFAR-10 [36] dataset, respectively. It is probably due to an overfitting problem when the depth is excessive. In fact, as a deep learning model, SAE is prone to overfitting. Specifically, as the number of layers increase, the number of network parameters that should be learned during the training phase increases. As such, it is essential to learn the optimal depth of an SAE that allows learning an abstract representation of the data while not being prone to overfitting. This is a motivation of the proposed approach which learns dynamically the depth while training the stacked AutoEncoder.

5.4. Experiment 2: Performance Assessment of the proposed approach DDSAE on Benchmark Datasets

The performance of the proposed approach DDSAE in dynamically learning the depth of an SAE in an unsupervised manner is assessed on the two benchmark datasets MNIST [35], and CIFAR-10 [36]. Initially, the number of AE layers is set to 20. After applying the proposed approach DDSAE, the initial model converged to a one-AE model on both datasets. Figure 6 displays the number of retained layers (the new learned depth) with respect to each iteration on both datasets MNIST [35] and CIFAR-10 [36]. The learned depth has progressively changed from 20 layers (in the first iteration) to one layer (after training 20 batches). Thus, the topology of the SAE changes dynamically while training the model from 20 AEs to one AE. This is consistent with the previous experimental finding.

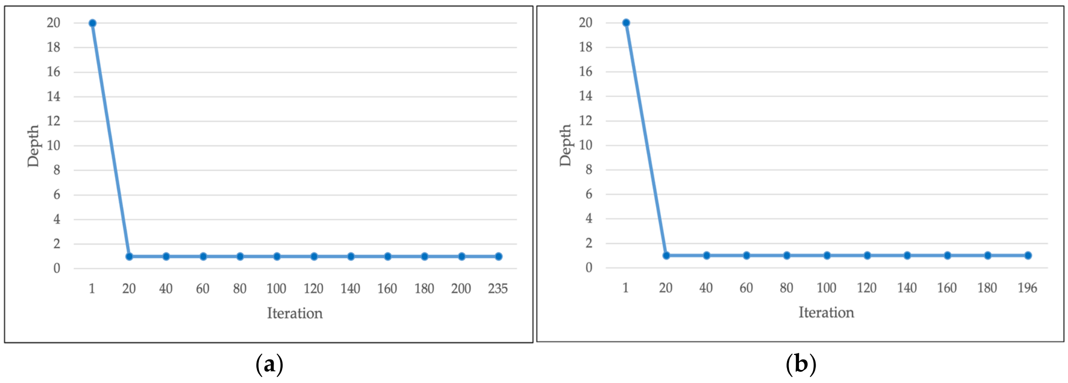


Figure 6. Number of retained layers with respect to each iteration on (**a**) MNIST dataset; and (**b**) CIFAR-10 dataset.

The steep fall in the depth depicted in Figure 6 reflects the fast convergence of the algorithm. In fact, after only 20 iterations, the proposed training approach is able to learn the optimal depth of the network.

Figure 7 displays the SAE loss function when training the model using MNIST [35] and CIFAR-10 [36] datasets. As can be seen, the loss function continues decreasing even after the convergence of the depth at iteration 20 (refer to Figure 6). Thus, the stacked AutoEncoder weights are progressively updated during the training. This excludes the possibility of saturation, especially that the employed cross-entropy loss function mitigates the effect of saturated output neurons [41].

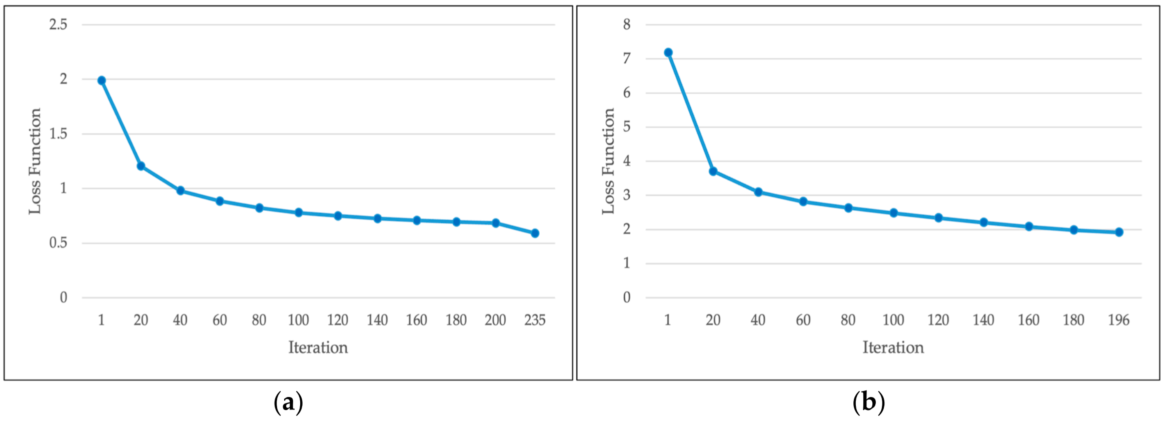


Figure 7. Loss function with respect to each iteration on (**a**) MNIST dataset; and (**b**) CIFAR-10 dataset.

The relevance layer weights are learned after training each batch using the update equation defined in Equation (3) subject to Equations (4) and (5). They represent the importance of each layer. In other words, they give insight on how much the feature map learned at a specific layer contributed to conserving the inner product between the input and the output. As such, they are used to prune irrelevant layers by discarding the layer exhibiting low relevance weights. Figures 8 and 9 display the layer relevance weight learned by DDSAE on MNIST [35] and CIFAR-10 [36] in the first and last batch, respectively.

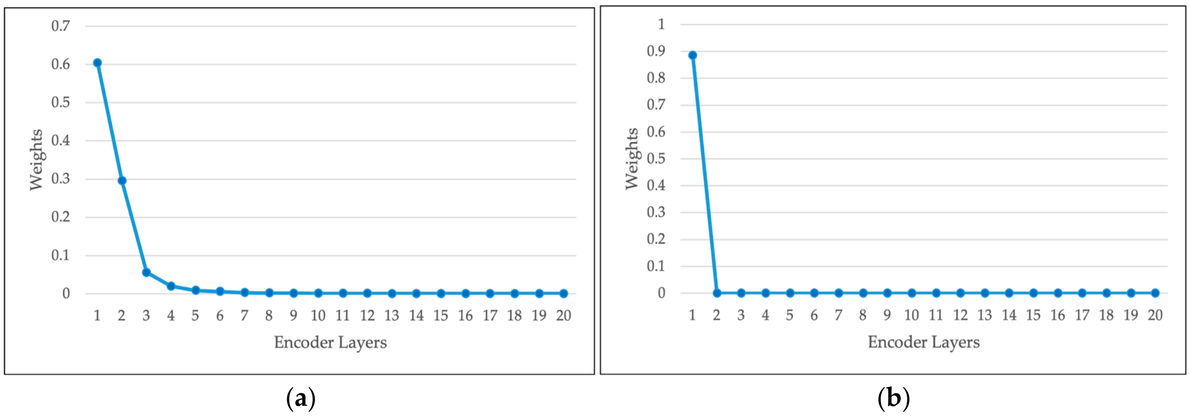


Figure 8. Layer relevance weights learned by DDSAE on MNIST dataset after training the model using (**a**) first batch; (**b**) last batch.

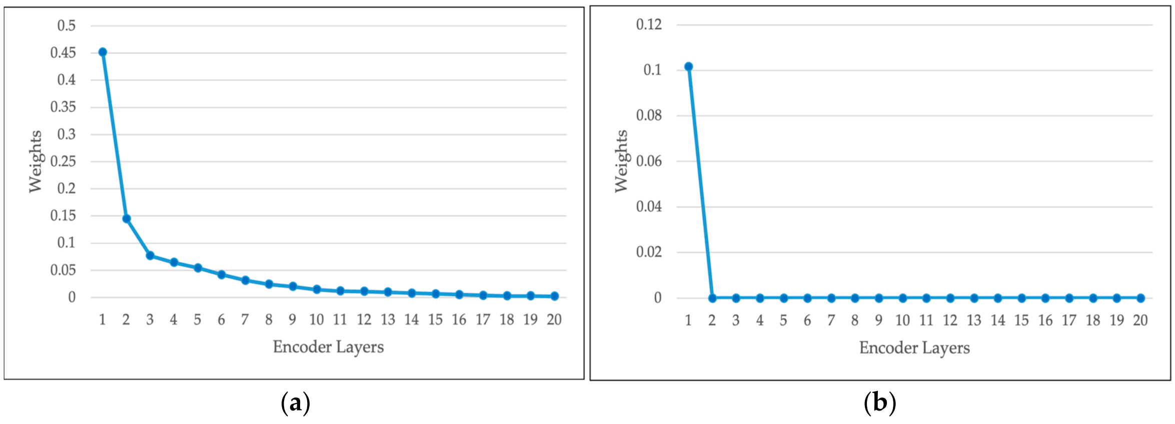


Figure 9. Layer relevance weights learned by DDSAE on CIFAR-10 dataset after training the model using (**a**) first batch; (**b**) last batch.

As can be observed, the maximum layer relevance weight in the first batch is assigned to the first layer of the encoder. Accordingly, starting from the last layer of the encoder, layers with relevance weight smaller than 75% of the maximum weight are pruned with their corresponding decoder layer. As a result, on MNIST [35] the number of trainable parameters has dropped from 41,609,784 to 3,571,784. Similarly, the trainable parameters on CIFAR-10 [36] have decreased from 46,188,072 to 8,150,072 after the algorithm has converged. This yields a lightweight model that reduces the risk of overfitting and enhances the model's generalization. Moreover, by learning the suitable relevance weight for each layer, the algorithm is able to determine the optimal depth for the models.

To measure the performance of the algorithm, various classifiers were adopted and appended on top of the encoder part. These classifiers are SoftMax layer, Support Vector Machine (SVM), and K-Nearest Neighbor (KNN) with number of neighbors equal to five. Table 4 reports the classification accuracy on the MNIST [35] and CIFAR-10 [36] datasets with respect to the considered classifiers. As can be observed in the obtained accuracies on MNIST [35], similar performance was achieved with respect to the three considered classifiers. However, the classification accuracy on CIFAR-10 [36] attained a better performance using the SoftMax classifier compared to SVM and KNN. Moreover, the classification accuracy achieved by DDSAE outperformed the classification accuracy of the manual depth investigation experiment. Specifically, the accuracy achieved by DDSAE on MNIST [35] is 97%, outperforming the one attained by manual investigation, which is 87.20%. Similarly, manual investigation performance reached 25.84% on CIFAR-10, while the performance of DDSAE reached an accuracy of 38%. Hence, while learning the optimal number of layers, DDSAE improved the performance of SAE.

Table 4. Accuracies achieved on MNIST and CIFAR-10 datasets.

Dataset		**MNIST**			**CIFAR-10**	
Classifier	SoftMax	SVM	KNN (k = 5)	SoftMax	SVM	KNN (k = 5)
Accuracy	97%	95.78%	95.93%	38%	33.39%	28.45%

5.5. Experiment 3: Performance Assessment of the Proposed Approach DDSAE on Real Datasets

To assess the performance of the proposed approach DDSAE in dynamically learning the depth of SAE, two real datasets, Parkinson's [37,38] and Breast Cancer [39,40], are considered. As in experiment 2, the initial number of AEs was set to 20. DDSAE converges to one AE layer for both datasets. These results are in accordance with the findings of experiment 2. Figure 10 shows the depth learned dynamically (the retained number of layers) while training each batch on the Parkinson's [37,38] and Breast Cancer [39,40] datasets. As can be seen, the depth decreased from 20 AEs (after training the first batch) to one AE (after training the second batch) for both datasets.

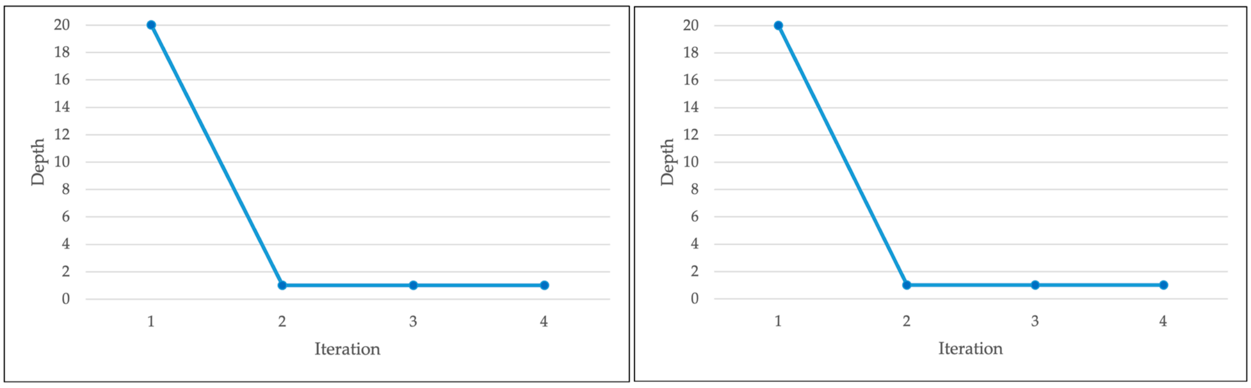
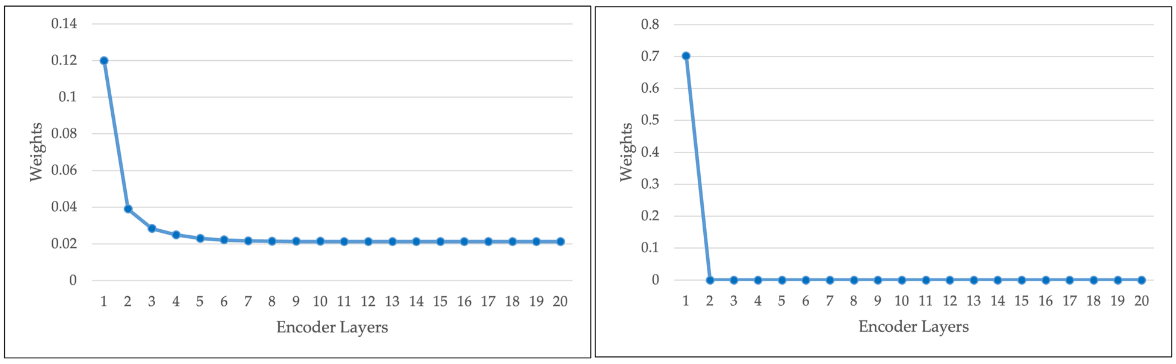


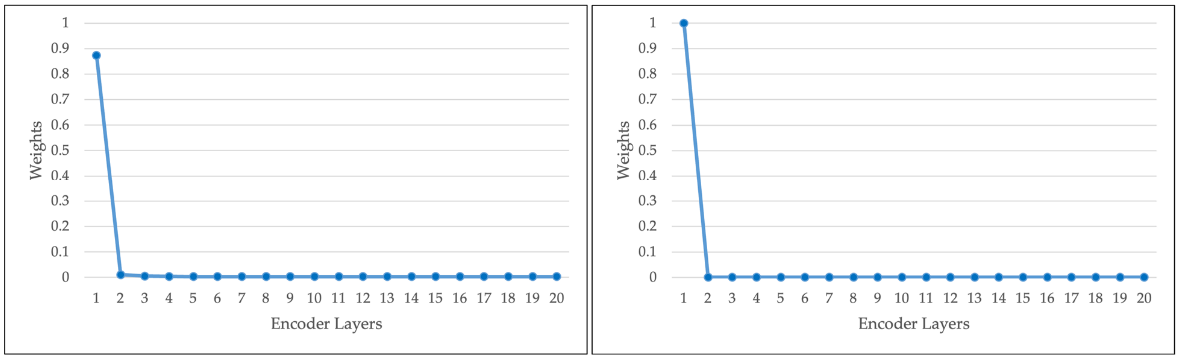
Figure 10. Number of retained layers with respect to each iteration on (**a**) Parkinson's dataset; and (**b**) Breast Cancer dataset.

During the training phase, the layer relevance weights are estimated in an unsupervised way at the end of each batch using the derived equation defined in Equation (3) subject to Equations (4) and (5). Giving insight into the importance of each layer, they are employed to dynamically prune irrelevant layers exhibiting low layer relevance weight. Figures 11 and 12 depict the layer relevance weights learned by DDSAE on the Parkinson's [37,38] and Breast Cancer [39,40] datasets in the first and last batch, respectively. As can be seen, after training the model using one batch, the maximum layer relevance weight is assigned to the first layer of the encoder. Hence, all layers exhibiting a relevance weight smaller than 75% of the maximum weight are pruned. In terms of learnable parameters, the number of learned parameters on Parkinson's [37,38] has dropped from 41,547,753 to 3,509,753 after the convergence of the model to one AE network. In addition, the learned parameters on Breast Cancer [39,40] decreased after convergence from 40,059,009 to 2,021,009 parameters.



(**a**) (**b**)

Figure 11. Layer relevance weights learned by DDSAE on Parkinson's dataset after training the model using (**a**) first batch; (**b**) last batch.



(**a**) (**b**)

Figure 12. Layer relevance weights learned by DDSAE on Breast Cancer dataset after training the model using (**a**) first batch; (**b**) last batch.

Moreover, the performance of DDSAE is assessed by appending different classifiers on top of the encoder part after training. Since the datasets are unbalanced, precision, recall, and F1-measure are reported in addition to the classification accuracy.

Table 5 reports the performance attained on the Parkinson's [37,38] and Breast Cancer [39,40] datasets using different classifiers. As can be observed, the performance on Parkinson's with the KNN and SVM classifiers achieved similar results; however, the SoftMax classifier achieved lower results. Similar results were obtained on the Breast Cancer dataset. Spherically, DDSAE reached an F1-score of 77% on the Parkinson's [37,38] dataset and an F1-score of 89% on the Breast Cancer [39,40] dataset when using the KNN classifier. Therefore, in addition to learning the optimal depth of the SAE, DDSAE is able to yield good classification results.

Table 5. Performance measures achieved on Parkinson's and Breast Cancer datasets.

Dataset	**Parkinson's**			**Breast Cancer**		
Classifier	SoftMax	SVM	KNN (k = 5)	SoftMax	SVM	KNN (k = 5)
Accuracy	75.77%	84.14%	85.02%	68.57%	82.86%	88.57%
Precision	87.7%	85.2%	84.3%	68.3%	82.7%	88.7%
Recall	52.5%	71.2%	74.0%	68.1%	82.7%	89.00%
F1-score	48.00%	74.6%	77.2%	68.1%	82.7%	89.00%

5.6. Experiment 4: Performance Comparison of DDSAE with State-of-the-Arts

In this experiment, we aim to compare the performance of the DDSAE algorithm to the state-of-the-art approaches which learn the depth of an SAE in an unsupervised manner. The considered approaches are the related works published in [26,27].

The work in [26], referred to as "Unsupervised restricted depth and width learning for a multi-layer AE", learns the width and depth of VAE using evolutionary search techniques. It is designed in such a way that the depth cannot exceed five layers and the width varies from 50 to 1000 nodes by a step of 50. The optimization of the depth or the width is achieved using the mutation operator and the fitness is defined as the inverse of the VAE reconstruction error. Similarly, the work in [27], called "Unsupervised width and depth learning using a chromosome of fixed length", is a genetic model which optimizes AE width and depth. Specifically, it employs crossover and mutation genetic operators on a 14-gene chromosome to create new candidate solutions. Particularly, the genes of the chromosome encode the number of layers (depth), the number of nodes per layer (width), the activation function, and the loss function. In the following, we refer to the works in [26,27] as "Approach1" and "Approach2", respectively.

The four datasets previously considered (MNIST [35], CIFAR-10 [36], Parkinson's [37,38], and Breast Cancer [39,40]) are used to assess the performance of the DDSAE approach and compare it to the state-of-the-art approaches. Moreover, the SoftMax classifier was appended on top of the encoder part after training the AE. Tables 6–9 depict the performance of DDSAE and compare it with the two state-of-the-art approaches on the four datasets along with the learned depth.

Table 6. Performance Comparison of DDSAE with state-of-the-art on MNIST.

Approach	Accuracy	F1-Score	Recall	Precision	Depth
DDSAE	**97%**	**97%**	**97%**	**97%**	1
Approach 1 [26]	75.98%	74.38%	75.16%	78.4%	1
Approach 2 [27]	91.26%	91.14%	91.13%	91.2%	1

Bold numbers indicate best performance.

Table 7. Performance Comparison of DDSAE with state-of-the-art on CIFAR-10.

Approach	Accuracy	F1-Score	Recall	Precision	Depth
DDSAE	**38%**	**36.5%**	**38%**	**38.6%**	1
Approach 1 [26]	20.99%	17.22%	20.99%	17.15%	1
Approach 2 [27]	32.02%	29.7%	32.02%	35.63%	1

Bold numbers indicate best performance.

Table 8. Performance Comparison of DDSAE with state-of-the-art on Parkinson.

Approach	Accuracy	F1-Score	Recall	Precision	Depth
DDSAE	75.77%	**48%**	52.5%	87.7%	1
Approach 1 [26]	50.22%	47%	50.14%	50.11%	5
Approach 2 [27]	74.45%	42.67%	50%	37.22%	2

Bold numbers indicate best performance.

Table 9. Performance Comparison of DDSAE with state-of-the-art on Breast Cancer.

Approach	Accuracy	F1-Score	Recall	Precision	Depth
DDSAE	68.57%	**68.1%**	68.1%	68.3%	1
Approach 1 [26]	48.57%	40.34%	45.72%	42.52%	4
Approach 2 [27]	54.29%	35.18%	50%	27.14%	1

Bold numbers indicate best performance.

As can be seen in Table 6, on the MNIST dataset [35] DDSAE achieved 97% with respect to the considered performance measures. Hence, it outperformed the first approach [26] by

21% and attained a better classification accuracy compared to the second approach [27]. All three approaches have converged to an architecture of one AE. Moreover, the results on the CIFAR-10 dataset [36], depicted in Table 7, show that DDSAE accuracy reached 38%. It was better than the first approach [26] by 17% and performed better than the second approach [27] by 6%. The depth obtained by the three considered approaches is one. The analysis of these results based on the accuracy metrics also applies to the other performance indicators depicted in Table 6, and 7, namely, F1-score, recall, and precision. The accuracy, F1-score, precision, and recall achieved by MNIST [35] and CIFAR-10 [36] are displayed in Figures 13 and 14, respectively.

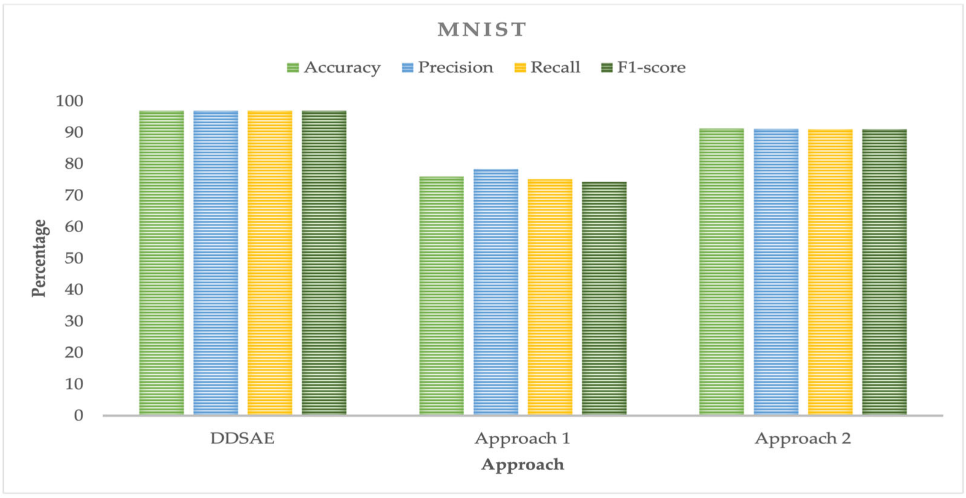


Figure 13. Performance comparison of DDSAE with state-of-the-art on MNIST.

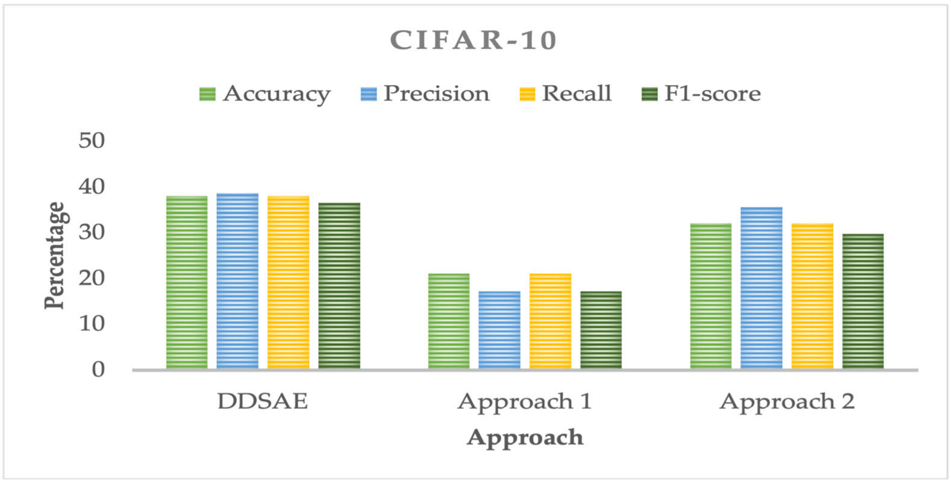


Figure 14. Performance comparison of DDSAE with state-of-the-art on CIFAR-10.

Furthermore, the performance of DDSAE on the Parkinson's [37,38] and Breast Cancer [39,40] datasets is depicted in Tables 8 and 9, respectively. The F1-score attained by DDSAE on the Parkinson's dataset is 48%. This result is slightly higher than the F1-score attained by the first approach [26], and is higher than the second approach [27] by 5%. Nevertheless, DDSAE converged to a single-layer model, whereas with the first approach [26] optimal depth was five layers of AE, and the second approach [27] reached two layers of AE models.

Finally, the performance on the Breast Cancer dataset [39,40] in terms of F1-score reached 68.1% when applying the DDSAE approach. This result is better than the two approaches in [26] and [27] by 27% and 33%, respectively. Overall, these results indicate that the DDSAE algorithm outperformed the state-of-the-art approaches that tend to learn the depth of an SAE in an unsupervised manner in terms of classification accuracy in balanced datasets, and in terms of F1-score in imbalanced datasets. Moreover, DDSAE outperformed the state-of-the-art approaches in the resulting architecture on the imbalanced datasets. The accuracy, F1-score, precision, and recall achieved on Parkinson's and Breast Cancer are displayed in Figures 15 and 16, respectively.

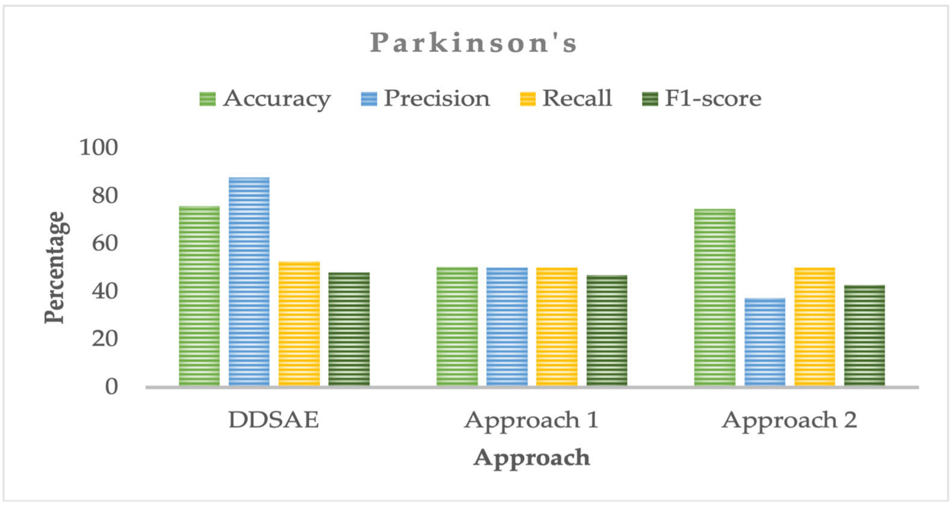


Figure 15. Performance comparison of DDSAE with state-of-the-art on Parkinson's.

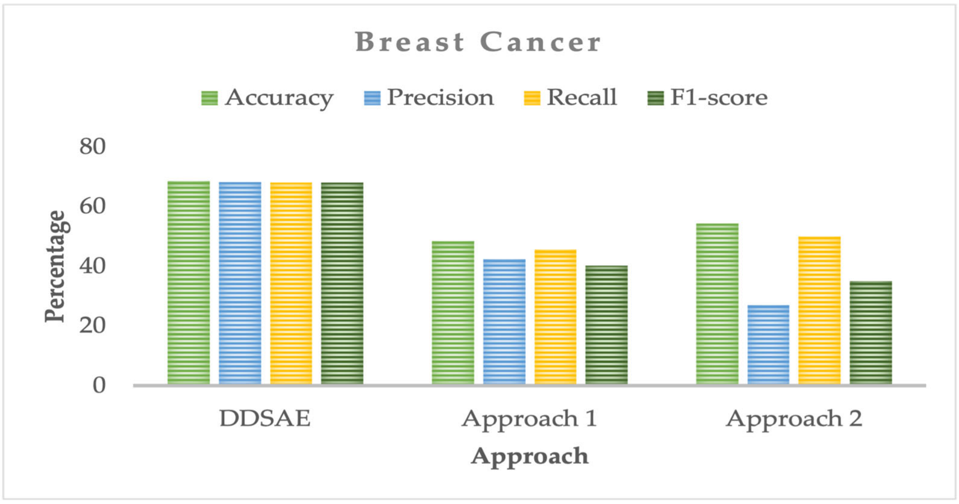


Figure 16. Performance comparison of DDSAE with state-of-the-art on Breast Cancer.

Real-world data exhibit generally overlapping and nonlinearly separable inter-class boundaries. This constitutes a challenge to machine learning models and thus affects drastically their performance. Accordingly, a nonlinear mapping can alleviate the problem. In particular, stacked AutoEncoders address the problem by automatically learning a nonlinear representation of the data. Specifically, they are trained in an unsupervised way to lean a meaningful representation of the data by optimizing the reconstruction error. However, their performance is drastically affected by their depth which impedes their broad usage. As such, learning the depth of the network while training it boosts the usage of AutoEncoder for real datasets.

Another significant comparison measure to report on is the testing time of the two approaches compared to the DDSAE algorithm. Table 10 depicts the testing time of all approaches on the four datasets. As can be observed, DDSAE is faster than the first and second approach. This is due to the fact that the DDSAE algorithm learned an optimal architecture with a reduced number of layers compared to the other approaches. As such, the depth plays an important role in the inference time.

Table 10. Testing time comparison with state-of-the-art.

Dataset	Testing Time (s)		
	DDSAE	**Approach 1 [26]**	**Approach 2 [27]**
MNIST	**1.99**	2.71	2.41
CIFAR-10	**2.11**	2.55	2.64
Parkinson's	**0.85**	1.49	1.47
Breast Cancer	**0.83**	1.34	1.37

Bold numbers indicate best performance.

However, in terms of training time, as depicted in Table 11, the DDSAE approach takes significantly longer than the state-of-the-art approaches. This is due to the fact that the proposed approach starts with an over-estimation of the number of layers and subsequently prunes non-relevant layers while training the model, whereas the learned optimal number of layers on the considered datasets converges to one layer. Nonetheless, since the training is done offline, it is insignificant for real-time applications that employ the previously trained model.

Table 11. Computational training time comparison with state-of-the-art.

Dataset	Computational Training Time (s)		
	DDSAE	**Approach 1 [26]**	**Approach 2 [27]**
MNIST	210	1.42	1.61
CIFAR-10	185	1.66	2
Parkinson's	22	6	3
Breast Cancer	22	3.51	1.53

6. Conclusions and Future Works

For the broad adoption of machine learning, it is crucial to learn data representation automatically and sidestep feature engineering. Thus, unsupervised representation leaning has emerged as an alternative solution to learn a representation of the data. Specifically, SAE has been employed for unsupervised mapping learning. It aims to learn a new mapping space by minimizing the reconstruction error. However, its performance proved to be susceptible to the model depth. As such, learning the topology of stacked AE plays an essential role in its performance. In fact, it controls the level of abstraction and complexity of the learned feature map. In this context, we proposed to learn the optimal depth of a stacked AE. This is done through learning the layers' relevance weights and while training the stacked AE. Specifically, we proposed to optimize a novel objective function apart from the loss function of the AE to learn the layers' relevance weights. These relevance weights will be used to prune irrelevant layers after training each batch, and thus dynamically update the architecture of the stacked AE. The experiments showed that DDSAE can dynamically learn the depth of a stacked AE in an unsupervised manner. Furthermore, DDSAE outperformed the state-of-the-art approaches on both benchmark and real datasets.

As future work, we intend to investigate the effect of the weight threshold ratio on the convergence of the training process in terms of computational time and resulting performance. Moreover, we plan to learn the optimal width in addition to the depth of a stacked AutoEncoder. This would be done by learning the relevance of each node at each layer, then pruning irrelevant nodes.

Author Contributions: Conceptualization, S.A., O.B. and M.M.B.I.; methodology, S.A., O.B. and M.M.B.I.; software, S.A.; validation, S.A. and O.B.; formal analysis, S.A. and O.B.; investigation, S.A. and O.B.; resources, S.A. and O.B.; data curation, S.A.; writing—original draft preparation, S.A.; writing—review and editing, O.B. and M.M.B.I.; visualization, S.A. and O.B.; supervision, O.B. and M.M.B.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. MNIST data can be found here: [http://yann.lecun.com/exdb/mnist/] (accessed on 1 November 2021); CIFAR-10 data can be found here: [https://www.cs.toronto.edu/~kriz/cifar.html] (accessed on 1 September 2022); Parkinson's dataset can be found here [https://archive.ics.uci.edu/ml/datasets/Parkinson%27s+Disease+Classification] (accessed on 1 January 2023); Breast Cancer dataset can be found here [https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra] (accessed on 1 January 2023).

Acknowledgments: The authors are grateful for the support of the Research Center of the College of Computer and Information Sciences, King Saud University. The authors thank the Deanship of Scientific Research and RSSU at King Saud University for their technical support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dargan, S.; Kumar, M.; Ayyagari, M.R.; Kumar, G. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. *Arch. Comput. Methods Eng.* **2020**, *27*, 1071–1092. [CrossRef]
2. Deng, L.; Li, X. Machine Learning Paradigms for Speech Recognition: An Overview. *IEEE Trans Audio Speech Lang Process* **2013**, *21*, 1060–1089. [CrossRef]
3. Shamim, S.M.; Miah, M.B.A.; Sarker, A.; Rana, M.; Jobair, A. Al Handwritten Digit Recognition Using Machine Learning Algorithms. *Indones. J. Sci. Technol.* **2018**, *3*, 29–39. [CrossRef]
4. Zhou, M.; Duan, N.; Liu, S.; Shum, H.Y. Progress in Neural NLP: Modeling, Learning, and Reasoning. *Engineering* **2020**, *6*, 275–290. [CrossRef]
5. Khan, A.I.; Al-Habsi, S. Machine Learning in Computer Vision. *Procedia Comput. Sci* **2020**, *167*, 1444–1451. [CrossRef]
6. Hatt, M.; Parmar, C.; Qi, J.; El Naqa, I. Machine (Deep) Learning Methods for Image Processing and Radiomics. *IEEE Trans. Radiat. Plasma Med Sci.* **2019**, *3*, 104–108. [CrossRef]
7. Dornadula, V.N.; Geetha, S. Credit Card Fraud Detection Using Machine Learning Algorithms. *Procedia Comput. Sci.* **2019**, *165*, 631–641. [CrossRef]
8. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Hasan, M.; Van Essen, B.C.; Awwal, A.A.S.; Asari, V.K. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **2019**, *8*, 292. [CrossRef]
9. Google AI. Available online: https://ai.google/tools/ (accessed on 8 March 2021).
10. Apple. Machine Learning Research. Available online: https://machinelearning.apple.com/ (accessed on 8 March 2021).
11. Facebook AI. Available online: https://ai.facebook.com/ (accessed on 8 March 2021).
12. Amazon AI. Available online: https://www.aboutamazon.com/news/amazon-ai (accessed on 8 March 2021).
13. Pawar, K.; Attar, V.Z. Assessment of Autoencoder Architectures for Data Representation. In *Deep Learning: Concepts and Architectures. Studies in Computational Intelligence*; Pedrycz, W., Chen, S.-M., Eds.; Springer: Cham, Switzerland, 2020; Volume 866, pp. 101–132.
14. Chicco, D.; Sadowski, P.; Baldi, P. Deep Autoencoder Neural Networks for Gene Ontology Annotation Predictions. In Proceedings of the ACM BCB 2014—5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, Newport Beach, CA, USA, 20–23 September 2014; pp. 533–540.
15. Alom, M.Z.; Taha, T.M. Network Intrusion Detection for Cyber Security Using Unsupervised Deep Learning Approaches. In Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 27–30 June 2017; pp. 63–69.
16. Zhang, G.; Liu, Y.; Jin, X. A Survey of Autoencoder-Based Recommender Systems. *Front. Comput. Sci.* **2020**, *14*, 430–450. [CrossRef]
17. Vareka, L.; Mautner, P. Stacked Autoencoders for the P300 Component Detection. *Front. Neurosci.* **2017**, *11*, 302. [CrossRef]
18. Chen, X.; Sun, Y.; Zhang, M.; Peng, D. Evolving Deep Convolutional Variational Autoencoders for Image Classification. *IEEE Trans. Evol. Comput.* **2020**, *25*, 815–829. [CrossRef]
19. Souza, P.V.d.C.; Torres, L.C.B.; Silva, G.R.L.; de Braga, A.P.; Lughofer, E. An Advanced Pruning Method in the Architecture of Extreme Learning Machines Using L1-Regularization and Bootstrapping. *Electronics* **2020**, *9*, 811. [CrossRef]

20. Palsson, B.; Sigurdsson, J.; Sveinsson, J.R.; Ulfarsson, M.O. Hyperspectral Unmixing Using a Neural Network Autoencoder. *IEEE Access* **2018**, *6*, 25646–25656. [CrossRef]
21. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
22. Zhai, J.; Zhang, S.; Chen, J.; He, Q. Autoencoder and Its Various Variants. In Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018; pp. 415–419.
23. Liu, G.; Bao, H.; Han, B. A Stacked Autoencoder-Based Deep Neural Network for Achieving Gearbox Fault Diagnosis. *Math. Probl. Eng.* **2018**, *2018*, 5105709. [CrossRef]
24. Gogoi, M.; Begum, S.A. Image Classification Using Deep Autoencoders. In Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2017, Coimbatore, India, 14–16 December 2017; pp. 1–5.
25. Xiong, Y.; Zuo, R. Recognition of Geochemical Anomalies Using a Deep Autoencoder Network. *Comput. Geosci.* **2016**, *86*, 75–82. [CrossRef]
26. Hajewski, J.; Oliveira, S. An Evolutionary Approach to Variational Autoencoders. In Proceedings of the 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 6–8 January 2020; pp. 71–77.
27. Charte, F.; Rivera, A.J.; Martínez, F.; del Jesus, M.J. Automating Autoencoder Architecture Configuration: An Evolutionary Approach. In Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation, Almería, Spain, 31 May–3 June 2019; Springer: Cham, Switzerland, 2019; pp. 339–349.
28. Lander, S.; Shang, Y. EvoAE-A New Evolutionary Method for Training Autoencoders for Deep Learning Networks. In Proceedings of the IEEE 39th Annual Computer Software and Applications Conference, Taichung, Taiwan, 1–5 July 2015; Volume 2, pp. 790–795.
29. Zhou, G.; Sohn, K.; Lee, H. Online Incremental Feature Learning with Denoising Autoencoders. *J. Mach. Learn. Res.* **2012**, *22*, 1453–1461.
30. Ashfahani, A.; Pratama, M.; Lughofer, E.; Ong, Y.S. DEVDAN: Deep Evolving Denoising Autoencoder. *Neurocomputing* **2020**, *390*, 297–314. [CrossRef]
31. Zhu, H.; Cheng, J.; Zhang, C.; Wu, J.; Shao, X. Stacked Pruning Sparse Denoising Autoencoder Based Intelligent Fault Diagnosis of Rolling Bearings. *Appl. Soft Comput.* **2020**, *88*, 106060. [CrossRef]
32. Cohoon, J.; Kairo, J.; Lienig, J. Evolutionary Algorithms for the Physical Design of VLSI Circuits. In *Advances in Evolutionary Computing: Theory and Applications*; Ghosh, A., Tsutsui, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 683–711.
33. Sun, S.; Chen, W.; Wang, L.; Liu, X.; Liu, T.Y. On the Depth of Deep Neural Networks: A Theoretical View. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
34. Saul, L.; Weinberger, K.; Ham, J.; Sha, F. Spectral Methods for Dimensionality Reduction. In *Semisupervised Learning*; MIT Press: Cambridge, MA, USA, 2006; Volume 3.
35. Yann, L.; Corinna, C.; Christopher, B. *The Mnist Database of Handwritten Digits*; The Courant Institute of Mathematical Sciences: New York, NY, USA, 1998.
36. Krizhevsky, A.; Nair, V.; Hinton, G. *CIFAR-10 and CIFAR-100 Datasets*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009.
37. Sakar, C.O.; Serbes, G.; Gunduz, A.; Tunc, H.C.; Nizam, H.; Sakar, B.E.; Tutuncu, M.; Aydin, T.; Isenkul, M.E.; Apaydin, H. A Comparative Analysis of Speech Signal Processing Algorithms for Parkinson's Disease Classification and the Use of the Tunable Q-Factor Wavelet Transform. *Appl. Soft Comput. J.* **2019**, *74*, 255–263. [CrossRef]
38. UCI. Machine Learning Repository: Parkinson's Disease Classification Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/Parkinson%27s+Disease+Classification (accessed on 24 April 2023).
39. Patrício, M.; Pereira, J.; Crisóstomo, J.; Matafome, P.; Gomes, M.; Seiça, R.; Caramelo, F. Using Resistin, Glucose, Age and BMI to Predict the Presence of Breast Cancer. *BMC Cancer* **2018**, *18*, 29. [CrossRef]
40. UCI. Machine Learning Repository: Breast Cancer Coimbra Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra (accessed on 24 April 2023).
41. Magnus, E. *Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, NLP, and Transformers Using TensorFlow*, 1st ed.; Deep Learning Institute: Santa Clara, CA, USA, 2021.