Autoencoders are unuspervised neural networks, meaning they are not trained with label information. Rather, the net is trained to reproduce its input (reconstruction), and the objective is usually something like minimizing mean squared error (MSE) of the reconstructed example vs the input example. They usually pass the input through hidden layers of smaller width than the input (encoding), resulting in some compression.

**Análisis de las máquinas “Sparse Autoencoders” como extractores de características:**

Los Auto-codiﬁcadores (Auto-Encoders, AE) son redes neuronales cuyo objetivo consiste en replicar los datos de entrada al sistema a la salida con la menor cantidad de distorsión posible. Los AE juegan un papel muy importante en el aprendizaje máquina (Machine Learning). Fueron introducidos inicialmente en los años 80 por Hinton y el grupo PDP para abordar el problema de "backpropagation sin profesor", usando los datos de entrada como "profesor". Junto con las reglas de aprendizaje Hebbiano, los AE conforman uno de los paradigmas

fundamentales para el aprendizaje no supervisado y para abordar el misterio de cómo los cambios sinápticos introducidos por eventos locales pueden coordinarse de forma auto-organizada para producir un aprendizaje global y un comportamiento inteligente. [11]

La Figura 2.1 muestra la arquitectura que presenta un AE. Se considera que

son simples MLP entrenados de forma no supervisada, puesto que las salidas

coinciden con los datos de entrada.

FIGURA 2.1: Arquitectura de un Auto-codiﬁcador

En la capa oculta del AE tiene lugar un "mapeo"de las características de en-

trada. La correspondiente codiﬁcación de los datos se obtiene a la salida de la

capa oculta, distinguiendo dos posibles situaciones: si el número de neuronas

ocultas (NH) es menor que la dimensión (D) de los datos de entrada NH < D,

resultará un código comprimido de los datos; mientras que si NH > D, se ob-

tiene una representación dispersa de los mismos.

Tal y como se ha introducido anteriormente, los auto-codiﬁcadores están

considerados como simples MLP entrenados de forma no supervisada (ver Sec-

ción 1.6.2 y 1.7.2), donde la dimensión de los datos de entrada coincide con la

de los datos de salida. En otras palabras, un auto-codiﬁcador consigue una ré-

plica de la entrada a la salida de su red. Lo realmente importante es que con el

auto-codiﬁcador se consigue que la red aprenda en su capa intermedia una re-

presentación de los datos de entrada sobre un espacio de diferente dimensión.

[7] En el caso que nos ocupa, la dimensión de los datos es inferior al conjunto de

entrada, con lo cual, el código resultante será una representación comprimida

de los mismos (ver Sección 2.1).

En nuestro proyecto, el auto-codiﬁcador en cuestión presenta la arquitectura

de la Figura 2.1. En ella distinguimos la capa de entrada (nodos color rojo), capa

oculta (nodos color verde) y la capa de salida (nodos color rojo). Además, pode-

mos distinguir los parámetros (W, b) = (W(1),W(2), b(1), b(2)), correspondientes

a los pesos y sesgos de la red, respectivamente y, donde (1) hace referencia a la

entrada de la capa oculta (l = 1) y (2) a la salida de la misma(l = 1). Del mismo

modo, denominaremos Wij a los pesos que conectan el nodo j en la capa l, con

el nodo i en la capa l+1. Además, la salida de la capa oculta se corresponde con

el valor que proporciona la función de activación, que denotaremos como a(l).

Esta función se corresponde con la función sigmoidal, que recordemos tiene la

siguiente ecuación.

f(x) = \frac{1}{1 + e^{-x}} (2.1)

De forma visual en la Figura 2.2 podemos ver la representación de las Ecuaciones (2.2 - 2.5) para la obtención de la señal de salida.

z(1) = W(1) · x + b(1)

a(2) =f(z(1))

z(2) = W(2) · a(2) + b(2)

y = z(2)

Las ecuaciones anteriormente descritas nos serán útiles para el entrenamiento del auto-codiﬁcador. En el caso que nos ocupa, el entrenamiento del auto-

codiﬁcador se realizará mediante el método de backpropagation, cuyo objetivo es el de minimizar la función de error que se obtiene a la salida de la red.

**Effective network intrusion detection via representation learning: A Denoising AutoEncoder approach:**

Autoencoder (AE) is an unsupervised-based neural network technique that represents the original data in a lower dimensionality and then reconstructs it into its original dimension [30–32]. Its architecture is composed of an encoder and decoder. The encoder employs a nonlinear transformation to map out the input data x to a hidden representation z, while the decoder uses the nonlinear transformation

to reconstruct ̂𝑥 from z.

Vincent et al. [33] proposed denoising autoencoder (DAE), a stochastic extension to the basic AE, designed to enhance the robustness of the original AE. DAE reconstructs the corrupted version of the

original data to its clean version. Through the stochastic mapping ̃𝑥 ∼𝑞 ( ̃𝑥|𝑥), the original input data x is corrupted to a partially destroyed input ̃𝑥. Next, the ̃𝑥 is propagated through DAE which are mapped

to the hidden representation z to finally be reconstructed as vector ̂𝑥.

Parameters of DAE are trained to reduce the total reconstruction error, and it adopt an encoding procedure extremely effective in retrieving

the representative features to neutralize the noise effect’s and achieve a better data reconstruction. DAE have attracted its application in low-dimensional data representation [34,35] and recover a noise-corrupted input data [36].

**Artificial Neural Network for Cybersecurity: A Comprehensive Review:**

B. Autoencoder

An unsupervised method is an autoencoder where the input is given as a vector. The network attempts to match and the output is the same as the input vector. One can generate a lower or higher dimensionality illustration of the data by getting the input and varying the recreating the input with its dimensionality. Data encoding operation (i.e., feature compression) is executed in the network with a small dimension of hidden layers. A denoising autoencoder can play an important role in order to eliminate the noise and reconstruct the original input from the noisy input. Figure 2 illustrates a basic autoencoder.

**Autoencoders; Dor Bank, Noam Koenigstein, Raja Giryes:**

Abstract An autoencoder is a speciﬁc type of a neural network, which is mainly

designed to encode the input into a compressed and meaningful representation, and

then decode it back such that the reconstructed input is similar as possible to the

original one.

Autoencoders have been ﬁrst introduced in [43] as a neural network that is trained

to reconstruct its input. Their main purpose is learning in an unsupervised manner

an “informative” representation of the data that can be used for various implications

such as clustering. The problem, as formally deﬁned in [2], is to learn the functions

𝐴 : R𝑛 → R𝑝 (encoder) and 𝐵 : R𝑝 → R𝑛 (decoder) that satisfy

arg min𝐴,𝐵 𝐸 [Δ(x, 𝐵 ◦ 𝐴(x)],

(1)

where 𝐸 is the expectation over the distribution of 𝑥, and Δ is the reconstruction loss

function, which measures the distance between the output of the decoder and the

intput. The latter is usually set to be the ℓ2-norm. Figure 1 provides an illustration of

the autoencoder model. In the most popular form of autoencoders, 𝐴 and 𝐵 are neural networks [40].

Since in training, one may just get the identity operator for 𝐴 and 𝐵, which keeps

the achieved representation the same as the input, some additional regularization is

required. The most common option is to make the dimension of the representation

smaller than the input. This way, a 𝑏𝑜𝑡𝑡𝑙𝑒𝑛𝑒𝑐𝑘 is imposed. This option also directly

serves the goal of getting a low dimensional representation of the data. This repre-

sentation can be used for purposes such as data compression, feature extraction, etc.

Its important to note that even if the 𝑏𝑜𝑡𝑡𝑙𝑒𝑛𝑒𝑐𝑘 is comprised of only one node, then

Autoencoders

3

overﬁtting is still possible if the capacity of the encoder and the decoder is large

enough to encode each sample to an index.

In cases where the size of the hidden layer is equal or greater than the size of the

input, there is a risk that the encoder will simply learn the identity function. To prevent

it without creating a bottleneck (i.e. smaller hidden layer) several options exists for

regularization, which we describe hereafter, that would enforce the autoencoder to

learn a diﬀerent representation of the input.

Denoising autoencoders [53] can be viewed either as a regularization option, or as

robust autoencoders which can be used for error correction. In these architectures,

the input is disrupted by some noise (e.g., additive white Gaussian noise or erasures

using Dropout) and the autoencoder is expected to reconstruct the clean version of

the input, as illustrated in Figure 2.

Learning a representation via the autoencoder can be used for various applications.

The diﬀerent types of autoencoders may be modiﬁed or combined to form new models for various applications. For example, in [39], they are used for classiﬁcation,

Autoencoders

9

captioning, and unsupervised learning. We describe below some of the applications

of autoencoders.

4.1 Autoencoders as a generative model

4.2 Use of autoencoders for classification

4.3 Use of autoencoders for clustering

4.4 Use of autoencoders for anomaly detection

4.5 Use of autoencoders for recommendation systems

4.6 Use of autoencoders for dimensionality reduction

**Detección de opiniones fraudulentas empleando Autoencoders**

. Nos centraremos en el uso de la técnica no supervisada denominada Autoencoder, que se ha utilizado en varios ámbitos tales como: generación de imágenes

[51], extracción de características [10], detección de opiniones fraudulentas [12] y

sistemas de recomendación [47].

El Autoencoder es un tipo de red neuronal artificial que se utiliza para aprender

codificaciones de datos eficientes de manera no supervisada. El principal objetivo

de un Autoencoder es aprender una representación (encoding) de un grupo de

datos, normalmente para la reducción de dimensionalidad. Aparte de la función

de reducción, está la función de reconstrucción, en donde el Autoencoder es capaz

de generar a partir de la codificación reducida una representación lo más cercana

posible al dato inicial.

En cuanto a la arquitectura que presentan los Autoencoders, destacamos que,

la forma más simple es una red neuronal no recurrente que tiene una capa de

entrada, una capa de salida y una o más capas ocultas que las conectan, con la

condición de que la capa de salida debe tener el mismo número de nodos que

la capa de entrada, con el propósito de reconstruir las entradas (Figura 2.4). Esta

arquitectura se podría descomponer en dos partes: una función encoder E = f(X)

que transforma las entradas X, en E codificadas; y una función decoder X0 = g(E)

que retorna una reconstrucción de las entradas X0

. Para aprender los pesos de las

11

Figura 2.4: Autoencoder

neuronas y, por tanto, las codificaciones, el Autoencoder busca minimizar alguna

función de pérdida, como el mean squared error (MSE), que penaliza a X0 por ser

diferente de X [6]:

L(x, x

0

) = |x − x

0

k

2 = kx − σ

0

(W0

(σ(Wx + b)) + b

0

)k

2

(1). Reducción de la dimensionalidad, (2). Procesamiento

de imágenes, (3). Interpretación automática, (4). Detección de anomalías

**Identificación de xenes relacionados con peor prognóstico en cancro de mama usando autoencoders:**

Los autoencoders son un tipo de redes de neuronas artificiales de aprendizaje no supervisado típicamente utilizados para reducir la dimensionalidad de los datos [15].

La arquitectura de un autoencoder puede dividirse en dos partes. En primer lugar el encoder, que codifica los datos de entrada en un nuevo vector de menor dimensión que el original.

Y en segundo lugar el decoder, que a partir del vector codificado trata de reconstruir la entrada

original (Figura 2.3). De este modo el autoencoder se ve forzado a preservar los aspectos más

relevantes de los datos de entrada para hacer una reconstrucción lo más óptima posible.

Figura 2.3: Esquema del funcionamiento de un autoencoder (Fuente [3]).

Estas arquitecturas permiten representar los datos con un número reducido de dimensiones y minimizando las pérdidas de reconstrucción. Sin embargo, no seleccionan un conjunto

de características presentes en los datos originales, y por lo tanto, no pueden utilizarse para

eliminar características redundantes y reducir costes. Una variante de los autoencoders son los

autoencoders discretos, diseñado para la selección de características.

**Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective:**

An auto-encoder (AE) [74] is a type of artifcial neural network used in an unsupervised way to learn efcient data

codes. The goal of an AE is to learn a representation for

a data set, typically by training the network to ignore the

‘noise’ signal for dimensionality reduction. An auto-encoder

consists of three components: encoder, code, and decoder

as shown in Fig. 6. The encoder compresses the input and

generates the data, and the decoder then uses this code to

reconstruct the input. One primary beneft of the AE is that

during propagation, this model can continuously extract useful features and flter the useless information [88]. A singlelayered AE with a linear activation function is very similar

to principal component analysis (PCA) [89], which is also

used to decrease the dimensionality of large data sets.

The auto-encoder is widely used for unsupervised learning tasks, e.g., dimension reduction, feature extraction,

efcient coding, and generative modeling [74, 90]. In the

domain of cybersecurity, the deep AE can be used to build

an efective security model. The reason is that the AE-based

feature learning model in cybersecurity typically uses the

minimum number of security features compared to other

state-of-the-art algorithms. The resulting rich and tiny latent

representation of the security features makes the model more

efective and efcient, even in small devices such as smartphones, known as the internet of things (IoT) devices [91].

For example, the authors [92] present an AE-based feature

learning model for cybersecurity applications, where they

have demonstrated the model efcacy for malware classifcation and detection of network-based anomalies. An

anomaly-based insider threat detection model using deep

AE has been presented in [93]. In [94], the authors present

a CNN-based android malware detection model, where they

use deep AE as a pre-training tool to minimize the time

of training. To enhance the intrusion detection method the

authors in [95] use a stacked sparse auto-encoder. Thus, the

AE-based model in the domain of cybersecurity can be useful due to its capability to capture the main features of data.

Auto-encoder (AE) An unsupervised learning algorithm that learns a representation ofthe

inputs and is deterministic

To signifcantly reduce the noise in the input data

Used typically for dimensionality reduction, very similar to PCA

**A Deep Auto-Encoder based Approach for Intrusion Detection System:**

An auto-encoder includes two parts: encoder and decoder.

Encoder aims to compress input data into a low-dimensional

representation, and decoder reconstruct input data based on

the low-dimension representation generated by the encoder.

On the other hand, auto-encoder can encode a representation

of an input layer into a hidden layer and then decode it into

an output layer [3].

For a given training dataset X = {x1, x2, ..., xm} with

m samples, where xi is a d-dimensional feature vector, the

encoder maps the input vector xi to a hidden representation

vector hi through a deterministic mapping fθ as given in (1)

hi = fθ(xi) = s(W xi + b), (1)

where W is a ´

d × d, ´

d is the number of hidden units, b is a

bias vector, θ is the mapping parameter set θ = {W, b}. s is

sigmoid activation function denoted as

s(t) = 1

1 + exp−t , (2)

where parameter t affects for the shape of the function.

The decoder maps back the resulting hidden representation

hi to a reconstructed d-dimensional vector yi in input space

as

yi = gθ

´(xi) = s(W h ´ i + ´b), (3)

where W´ is a d × ´

d, ´b is a bias vector and ´θ = {W , ´ ´b}.

The goal of training the autoencoder is to minimize the

difference between input and output. Therefore, a loss function

is calculated by the following equation

L(x, y) = 1

m

m

i=1

xi − yi

2 , (4)

where m is the total number of training dataset.

The main objective is to find the optimal parameters (θ and ´θ) which can be effectively minimize the difference between

input and reconstructed output over the whole training set as

θ = {W, b} = argθminL(x, y).

**Effective network intrusion detection via representation learning: A Denoising AutoEncoder approach:**

An autoencoder is employed for representation learning in this study to reduce dataset dimensionality and extract

meaningful features.

AutoEncoder is an

unsupervised learning technique, while the tabular model belongs to

the supervised learning category.

Autoencoder (AE) is an unsupervised-based neural network technique that represents the original data in a lower dimensionality and

then reconstructs it into its original dimension [30–32]. Its architecture is composed of an encoder and decoder. The encoder employs

a nonlinear transformation to map out the input data x to a hidden

representation z, while the decoder uses the nonlinear transformation

to reconstruct ̂𝑥 from z.

Vincent et al. [33] proposed denoising autoencoder (DAE), a

stochastic extension to the basic AE, designed to enhance the robustness of the original AE. DAE reconstructs the corrupted version of the

original data to its clean version. Through the stochastic mapping ̃𝑥 ∼

𝑞 (̃𝑥|𝑥), the original input data x is corrupted to a partially destroyed

input ̃𝑥. Next, the ̃𝑥 is propagated through DAE which are mapped

to the hidden representation z to finally be reconstructed as vector ̂𝑥.

Parameters of DAE are trained to reduce the total reconstruction error,

and it adopt an encoding procedure extremely effective in retrieving

the representative features to neutralize the noise effect’s and achieve

a better data reconstruction. DAE have attracted its application in lowdimensional data representation [34,35] and recover a noise-corrupted

input data [36].

**Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study:**

The deep auto-encoder was used by Shone et al. (2018)

[59] for cyber security intrusion detection. They use an autoencoder featuring non-symmetrical multiple hidden layers to

facilitate improved classification results compared with deep

belief networks. The study uses the KDD Cup ’99 and NSLKDD datasets with five metrics performances, including, accuracy, precision, recall, false alarm, and F-score. The results on

the KDD Cup ’99 dataset evaluation show that the proposed

model is able to offer an average accuracy of 97.85%, which

is better compared to the work in [45]. In addition, the results

on the NSL-KDD dataset evaluation show that the proposed

model offered a total accuracy rate of 85.42%, which is an

improvement upon the deep belief network model by 5%.

Khan et al. [60] proposed an intrusion detection system

based on the two-stage deep learning model, named TSDL.

The TSDL model uses a stacked auto-encoder with a soft-max

classifier, which is composed of three main layers, namely, 1)

the input layer, 2) the hidden layers, and 3) the output layer.

These three layers employ a feed-forward neural network

similar to a multi-layer perceptron. The study uses two public

datasets, including, KDD99 and UNSW-NB15 datasets. The

results on KDD99 dataset achieve high recognition rates, up

to 99.996%. In addition, the results on UNSW-NB15 dataset

achieve high recognition rates, up to 89.134%.

To design a self-adaptive and autonomous misuse intrusion

detection system, Papamartzivanos et al. [61] propose the use

of auto-encoder techniques. Specifically, the proposed system

is based in four phases, including 1) Monitor, 2) Analyze,

3) Plan, 4) Execute, and 5) Knowledge. The monitor phase

determines any alteration event that requires an intrusion

detection system adaptation. The analyze phase uses network

audit tools (e.g., Argus and CICFlowMeter) to perform the

transformation of the raw network traffic into network flows.

The plan phase uses a sparse autoencoder to learn representations of the input data. The execute phase is accountable

for storing purposes. However, the study uses two datasets

in performance evaluation, including KDDCup’99 and NSLKDD. The results show that the average accuracy of the static

model is 59.71% while for the adaptive model it is 77.99%

The combination of an improved conditional variational

autoencoder and deep neural network was used by Yang et

al. [62] for cyber security intrusion detection. The proposed

study consists of three phases: 1) training, 2) generating new

attacks and 3) detecting attacks. The training phase consists

of optimizing the loss of the encoder and the decoder. The

phase of generating new attacks uses a multivariate Gaussian

distribution as the distribution. The phase of detecting attacks

employs a deep neural network to detect attacks. To validate

the proposed model, the NSL-KDD and UNSW-NB15 datasets

are used, which the default learning rate of the Adam optimizer

is 0.001. The results show the highest accuracy of 89.08% and

detection rate of 95.68% on the UNSW-NB15 dataset.

To construct a deep neural network, Abusitta et al. [63]

uses a denoising autoencoder as a building block for cyber

security intrusion detection. The denoising autoencoder is used

to learning how to reconstruct intrusion detection systems

feedback from partial feedback. The KDD Cup 99 dataset is

used on the performance evaluation, which the results show

that the proposed model can achieve detection accuracy up

to 95%. Therefore, stacked denoising auto-encoders is used

by Wang et al. (2016) [64] for detecting malicious JavaScript

code. The study uses a dataset, which is composed of 12 320

benign and 14 783 malicious JavaScript samples. With three

layers of auto-encoders and 250 hidden units, the experimental

results show an optimal choice for building an intrusion

detection system based on the deep learning technique.

4) Deep auto encoders (DA): An autoencoder is composed

of both the encoder and the decoder, as presented in Figure 9.

Refer to [151], these two parts can be defined as follow:

encoderG (x) = s(W x + b) (10)

decoderG0 (y) = s(W0

y + b

0

) (11)

where G = {W, b}; G

0 = {W0

, b

0

}; W is a d

0 × d weight

matrix; x is an input vector; y is the hidden representation; b

is an offset vector of dimensionality d

0.

**A deep learning approach for proactive multi-cloud cooperative intrusion detection system:**

3.2. The traditional autoencoders

An autoencoder is an unsupervised learning approach that is

used to learn efficient and reliable data codings [31]. It is used

to pre-train each layer in a deep neural network in order to

obtain better initial weights that lead to a better performing classification [8]. Researchers have seen that weights initialization

using autoencoders can improves the performance of deep neural

networks compared to random initialization [8].

The structure of an autoencoder is shown in Fig. 2. The dimensions for both input (IDSs’ feedback) and output (reconstruction

of IDSs’ feedback) are the same as shown in the figure. Labels

(i.e., +1) in Fig. 2 represent a bias vector. The use of biases in

a neural network increases the capacity of the network to solve

problems [32].

An autoencoder is used as a building block for deep networks [8]. In particular, it takes input vector (IDSs’ feedback)

x ∈ [0, 1]

d

, where d is the vector dimension, and maps it to a

hidden representation h ∈ [0, 1]

d

′

using the following equation:

h = fθ (x) = Sigmoid(W ∗ x + b) (1)

A. Abusitta, M. Bellaiche, M. Dagenais et al. / Future Generation Computer Systems 98 (2019) 308–318 311

Fig. 2. Example of an autoencoder.

θ = {W, b}, W is a weight matrix and b is a bias vector.

After that, the resulting hidden layer representation h will be

reconstructed to the output layer x

′ using a decoding function as

follows:

x

′ = gθ

′(h) = Sigmoid(W′

∗ h + b

′

) (2)

θ

′ = {W′

, b

′

}, W′

and b

′

are a weight matrix and a bias

vector of the reverse mapping, respectively. The weight matrix

W′ of the reverse mapping may optionally be constrained by

W′ = WT

, in which case the autoencoder is said to have tied

weights [8,10]. Each training x is thus mapped to a corresponding

h and a reconstruction x

′

.

The purpose of the model is to optimize the parameters (θ

and θ

′

) of the model, so that the reconstruction error between

input and output can be minimized. The following optimization

problem is used for this purpose:

θ

∗

, θ′∗ = arg minθ ,θ′

1

n

∑n

i=1

L(x

(i)

, x

′(i)

)

=

1

n

∑n

i=1

L(x

(i)

, gθ

′(fθ (x

(i)

)))

(3)

where L is a loss function. In an unsupervised learning model, a

loss function is a measure of how close the reconstructed input

is to the original input [32].

Since the input vector x is a binary vector (an IDS’s feedback

is either 0 or 1), a reconstruction cross-entropy is used as a

loss function [33]. Thus, the loss function will be described as

follows:

L(x, x

′

) = −∑

d

i=1

[xi

logx′

i + (1 − xi)log(1 − x

′

i

)]

The traditional autoencoder is then used to take corrupted

data z and try to learn how to reconstruct x. This is done by

allowing the input z to be mapped to a hidden representation as:

h = fθ (z) = Sigmoid(W′

∗ z + b

′

) (6)

Note that we selected z as input instead of x as a traditional

autoencoder was used. The result of the above equation h is then

used to reconstruct x

′

as follows:

x

′ = gθ

′(h) = Sigmoid(W ∗ h + b)

**A Hybrid Spectral Clustering and Deep Neural Network Ensemble Algorithm for Intrusion Detection in Sensor Networks:**

2.2.1. Auto-Encoders

An auto-encoder is a type of unsupervised three-layer neural network whose output target

is input data [34], which is described in Figure 1. The auto-encoder includes both encoder and

decoder networks. The encoder network transforms input data from a high-dimensional space to code

in a low-dimensional space, and the decoder network remodels this input from the previous step.

The encoder network is deﬁned as an encoding function denoted by fencoder. This function details the

encoding process:

hm = fencoder(xm)

(2)

where xm is a data point and hm is the encoding vector obtained from xm .

2.2.2. Decoder

The decoder network is deﬁned as a reconstruction function denoted as fdecoder and is described

as follows:

xˆm = fdecoder(hm)

(3)

where xm is the decoding vector obtained from hm. T

2.2.4. Denoising Auto-Encoders (DAEs)

There is a way to capture something useful about the hidden units with input data. DAEs are used

to discover more robust hidden-layer features and can effectively prevent the networks from simply

learning features. The DAE is a stochastic auto-encoder that tries to represent information about the

input data that can only be learned by capturing the statistical attributes between the encoded and

input data. The DAEs can be represented in a variety of ways, including from stochastic operator,

bottom-up-information and top-down-generative model perspectives. In this paper, the DAEs learned

feature representations by adjusting partial corruption from the input pattern [39]. A ﬁxed number of

components in each input x(i) are chosen randomly and their value set to zero; this distribution is

called qD(·), then the joint distribution of input and output is deﬁned as follows:

q0(X, χ, y) = q0(X)qD(χ|X)δfθ(χ)(y)

(15)

in which the encoding of X is a stochastic mapping by the distribution χ ∼ qD(χ|X). If the condition

fθ(x) = y · q0(X) is satisﬁed, then δfθ(χ)(y) indicates the empirical distribution associated with the

input data and set to 0. χ can be determined by y. In order to obtain the best reconstructed version of

X with consideration between χ and y, let θ be a parameter of the joint distribution of Equation (15),

and calculate the minimum error of the cost function of the input matrix X from y as follows:

arg min

θ,θ Eq0(X,χ)[L(X, X)]

(16)

Sensors 2016, 16, 1701

7 of 23

where [L(X, X)] is the error function depending on input x and the reconstructed X. The best cost

function value is then obtained using SGD. To encode optimised denoising codes, there is a trick

in encoder process. Let y be a hidden representation obtained with the function sigmoid(Wχ + b),

where χ is a generated corrupted version of the input data and the parameters of the function are

denoted as θ = {W, b}. Each corrupted code contains some elements of input X randomly set to 0.

Then, the ﬁnal corrupted version can be minimised in Equation (16).

**A Hybrid Malicious Code Detection Method based on Deep Learning:**

2. 1 AutoEncoder Dimensionality Reduction

AutoEncoder [14] is a kind of deep learning method for learning efficient code which is

proposed by G. E. Hinton in 2006. Through the study of the compression coding of

specified set of data, it can achieve the purpose of data dimensionality reduction.

AutoEncoder structure is divided into part of encoder and decoder, including input layer,

hidden layer, output layer. The cross section between encoder and decoder named code

layer is the core of AutoEncoder that can reflect the essential characteristics of high

dimensional data set with nested structure, and to set the intrinsic dimensions of

high-dimensional data sets. When the number of hidden layer neurons are less than the

number of input layer and output layer neurons, we can get the compressed vector of input

layer called the data dimensionality reduction.

AutoEncoder consists of three steps, which are pretraining, Unrolling and fine-tuning

process [14], as shown in Figure 1.

**Análisis de las máquinas “Sparse**

**Autoencoders” como extractores de**

**características:**

Los Auto-Codificadores conforman la base del trabajo desarrollado. Es por

ello que el presente capítulo ahonda en su estructura y arquitectura, así como

en los Auto-Codificadores Dispersos.

2.1. Introducción

Los Auto-codificadores (Auto-Encoders, AE) son redes neuronales cuyo objetivo

consiste en replicar los datos de entrada al sistema a la salida con la menor

cantidad de distorsión posible. Los AE juegan un papel muy importante en el

aprendizaje máquina (Machine Learning). Fueron introducidos inicialmente en

los años 80 por Hinton y el grupo PDP para abordar el problema de "backpropagation

sin profesor", usando los datos de entrada como "profesor". Junto con

las reglas de aprendizaje Hebbiano, los AE conforman uno de los paradigmas

fundamentales para el aprendizaje no supervisado y para abordar el misterio

de cómo los cambios sinápticos introducidos por eventos locales pueden coordinarse

de forma auto-organizada para producir un aprendizaje global y un

comportamiento inteligente. [11]

La Figura 2.1 muestra la arquitectura que presenta un AE. Se considera que

son simples MLP entrenados de forma no supervisada, puesto que las salidas

coinciden con los datos de entrada.

FIGURA 2.1: Arquitectura de un Auto-codificador

En la capa oculta del AE tiene lugar un "mapeo"de las características de entrada.

La correspondiente codificación de los datos se obtiene a la salida de la

capa oculta, distinguiendo dos posibles situaciones: si el número de neuronas

ocultas (NH) es menor que la dimensión (D) de los datos de entrada NH < D,

resultará un código comprimido de los datos; mientras que si NH > D, se obtiene

una representación dispersa de los mismos. [12]

18 Capítulo 2. Auto-Codificadores

A continuación, se procede a describir la estructura del AE, explicando los

algoritmos de aprendizaje empleados para el estudio realizado, así como la estructura

del AE de Dispersión (Sparse AE).

2.2. Auto-Codificador

Tal y como se ha introducido anteriormente, los auto-codificadores están

considerados como simples MLP entrenados de forma no supervisada (ver Sección

1.6.2 y 1.7.2), donde la dimensión de los datos de entrada coincide con la

de los datos de salida. En otras palabras, un auto-codificador consigue una réplica

de la entrada a la salida de su red. Lo realmente importante es que con el

auto-codificador se consigue que la red aprenda en su capa intermedia una representación

de los datos de entrada sobre un espacio de diferente dimensión.

[7] En el caso que nos ocupa, la dimensión de los datos es inferior al conjunto de

entrada, con lo cual, el código resultante será una representación comprimida

de los mismos (ver Sección 2.1).

En nuestro proyecto, el auto-codificador en cuestión presenta la arquitectura

de la Figura 2.1. En ella distinguimos la capa de entrada (nodos color rojo), capa

oculta (nodos color verde) y la capa de salida (nodos color rojo). Además, podemos

distinguir los parámetros (W; b) = (W(1);W(2); b(1); b(2)), correspondientes

a los pesos y sesgos de la red, respectivamente y, donde (1) hace referencia a la

entrada de la capa oculta (l = 1) y (2) a la salida de la misma(l = 1). Del mismo

modo, denominaremos Wij a los pesos que conectan el nodo j en la capa l, con

el nodo i en la capa l+1. Además, la salida de la capa oculta se corresponde con

el valor que proporciona la función de activación, que denotaremos como a(l).

Esta función se corresponde con la función sigmoidal, que recordemos tiene la

siguiente ecuación.

**Artificial Neural Network for Cybersecurity: A Comprehensive Review:**

B. Autoencoder

An unsupervised method is an autoencoder where the input is given as a vector. The network attempts to match and the output is the same as the input vector. One can generate a lower or higher dimensionality illustration of the data by getting the input and varying the recreating the input with its dimensionality. Data encoding operation (i.e., feature compression) is executed in the network with a small dimension of hidden layers. A denoising autoencoder can play an important role in order to eliminate the noise and reconstruct the original input from the noisy input. Figure 2 illustrates a basic autoencoder.

**Autoencoders; Dor Bank, Noam Koenigstein, Raja Giryes:**

Abstract An autoencoder is a specific type of a neural network, which is mainly

designed to encode the input into a compressed and meaningful representation, and

then decode it back such that the reconstructed input is similar as possible to the

original one.

Autoencoders have been first introduced in [43] as a neural network that is trained

to reconstruct its input. Their main purpose is learning in an unsupervised manner

an “informative” representation of the data that can be used for various implications

such as clustering. The problem, as formally defined in [2], is to learn the functions

𝐴 : R

𝑛 → R

𝑝

(encoder) and 𝐵 : R

𝑝 → R

𝑛

(decoder) that satisfy

arg min𝐴,𝐵 𝐸[Δ(x, 𝐵 ◦ 𝐴(x)], (1)

where 𝐸 is the expectation over the distribution of 𝑥, and Δ is the reconstruction loss

function, which measures the distance between the output of the decoder and the

intput. The latter is usually set to be the ℓ2-norm. Figure 1 provides an illustration of

the autoencoder model. In the most popular form of autoencoders, 𝐴 and 𝐵 are neural networks [40]. In

the special case that 𝐴 and 𝐵 are linear operations, we get a linear autoencoder [3].

In the case of linear autoencoder where we also drop the non-linear operations, the

autoencoder would achieve the same latent representation as Principal Component

Analysis (PCA) [38]. Therefore, an autoencoder is in fact a generalization of PCA,

where instead of finding a low dimensional hyperplane in which the data lies, it is

able to learn a non-linear manifold.

Autoencoders may be trained end-to-end or gradually layer by layer. In the latter

case, they are ”stacked” together, which leads to a deeper encoder. In [35], this

is done with convolutional autoencoders, and in [54] with denoising autoencoder

(described below).

Denoising autoencoders [53] can be viewed either as a regularization option, or as

robust autoencoders which can be used for error correction. In these architectures,

the input is disrupted by some noise (e.g., additive white Gaussian noise or erasures

using Dropout) and the autoencoder is expected to reconstruct the clean version of

the input, as illustrated in Figure 2.

Fig. 2: A denoising autoencoder example. The disrupted input image is encoded to

a representation and then decoded.

Note that x˜ is a random variable, whose distribution is given by 𝐶(x˜|x). Two

common options for 𝐶 are:

𝐶𝜎 (x˜|x) = N (x, 𝜎2I), (4)

and

𝐶𝑝 (x˜|x) = 𝛽 , x, 𝛽 ∼ 𝐵𝑒𝑟(𝑝), (5)

where detnotes an element-wise (Hadamard) product. In the first option, the

variance parameter 𝜎 sets the impact of the noise. In the second, the parameter

𝑝 sets the probability of a value in x not being nullified. A relationship between

denoising autoencoders with dropout to analog coding with erasures has been shown

in [4].

4.1 Autoencoders as a generative model

4.2 Use of autoencoders for classification

4.3 Use of autoencoders for clustering

4.4 Use of autoencoders for anomaly detection

4.5 Use of autoencoders for recommendation systems

4.6 Use of autoencoders for dimensionality reduction

**Detección de opiniones fraudulentas**

**empleando Autoencoders**

. Nos centraremos en el uso de la técnica no supervisada denominada Autoencoder, que se ha utilizado en varios ámbitos tales como: generación de imágenes

[51], extracción de características [10], detección de opiniones fraudulentas [12] y

sistemas de recomendación [47].

El Autoencoder es un tipo de red neuronal artificial que se utiliza para aprender

codificaciones de datos eficientes de manera no supervisada. El principal objetivo

de un Autoencoder es aprender una representación (encoding) de un grupo de

datos, normalmente para la reducción de dimensionalidad. Aparte de la función

de reducción, está la función de reconstrucción, en donde el Autoencoder es capaz

de generar a partir de la codificación reducida una representación lo más cercana

posible al dato inicial.

En cuanto a la arquitectura que presentan los Autoencoders, destacamos que,

la forma más simple es una red neuronal no recurrente que tiene una capa de

entrada, una capa de salida y una o más capas ocultas que las conectan, con la

condición de que la capa de salida debe tener el mismo número de nodos que

la capa de entrada, con el propósito de reconstruir las entradas (Figura 2.4). Esta

arquitectura se podría descomponer en dos partes: una función encoder E = f(X)

que transforma las entradas X, en E codificadas; y una función decoder X0 = g(E)

que retorna una reconstrucción de las entradas X0

. Para aprender los pesos de las

11

Figura 2.4: Autoencoder

neuronas y, por tanto, las codificaciones, el Autoencoder busca minimizar alguna

función de pérdida, como el mean squared error (MSE), que penaliza a X0 por ser

diferente de X [6]:

L(x, x

0

) = |x − x

0

k

2 = kx − σ

0

(W0

(σ(Wx + b)) + b

0

)k

2

(1). Reducción de la dimensionalidad, (2). Procesamiento

de imágenes, (3). Interpretación automática, (4). Detección de anomalías:

**Identificación de xenes relacionados con**

**peor prognóstico en cancro de mama**

**usando autoencoders**

Los autoencoders son un tipo de redes de neuronas artificiales de aprendizaje no supervisado típicamente utilizados para reducir la dimensionalidad de los datos [15].

La arquitectura de un autoencoder puede dividirse en dos partes. En primer lugar el encoder, que codifica los datos de entrada en un nuevo vector de menor dimensión que el original.

Y en segundo lugar el decoder, que a partir del vector codificado trata de reconstruir la entrada

original (Figura 2.3). De este modo el autoencoder se ve forzado a preservar los aspectos más

relevantes de los datos de entrada para hacer una reconstrucción lo más óptima posible.

Figura 2.3: Esquema del funcionamiento de un autoencoder (Fuente [3]).

Estas arquitecturas permiten representar los datos con un número reducido de dimensiones y minimizando las pérdidas de reconstrucción. Sin embargo, no seleccionan un conjunto

de características presentes en los datos originales, y por lo tanto, no pueden utilizarse para

eliminar características redundantes y reducir costes. Una variante de los autoencoders son los

autoencoders discretos, diseñado para la selección de características.

**Deep Cybersecurity: A Comprehensive Overview from Neural Network**

**and Deep Learning Perspective**

An auto-encoder (AE) [74] is a type of artifcial neural network used in an unsupervised way to learn efcient data

codes. The goal of an AE is to learn a representation for

a data set, typically by training the network to ignore the

‘noise’ signal for dimensionality reduction. An auto-encoder

consists of three components: encoder, code, and decoder

as shown in Fig. 6. The encoder compresses the input and

generates the data, and the decoder then uses this code to

reconstruct the input. One primary beneft of the AE is that

during propagation, this model can continuously extract useful features and flter the useless information [88]. A singlelayered AE with a linear activation function is very similar

to principal component analysis (PCA) [89], which is also

used to decrease the dimensionality of large data sets.

The auto-encoder is widely used for unsupervised learning tasks, e.g., dimension reduction, feature extraction,

efcient coding, and generative modeling [74, 90]. In the

domain of cybersecurity, the deep AE can be used to build

an efective security model. The reason is that the AE-based

feature learning model in cybersecurity typically uses the

minimum number of security features compared to other

state-of-the-art algorithms. The resulting rich and tiny latent

representation of the security features makes the model more

efective and efcient, even in small devices such as smartphones, known as the internet of things (IoT) devices [91].

For example, the authors [92] present an AE-based feature

learning model for cybersecurity applications, where they

have demonstrated the model efcacy for malware classifcation and detection of network-based anomalies. An

anomaly-based insider threat detection model using deep

AE has been presented in [93]. In [94], the authors present

a CNN-based android malware detection model, where they

use deep AE as a pre-training tool to minimize the time

of training. To enhance the intrusion detection method the

authors in [95] use a stacked sparse auto-encoder. Thus, the

AE-based model in the domain of cybersecurity can be useful due to its capability to capture the main features of data.

Auto-encoder (AE) An unsupervised learning algorithm that learns a representation ofthe

inputs and is deterministic

To signifcantly reduce the noise in the input data

Used typically for dimensionality reduction, very similar to PCA

**A Deep Auto-Encoder based Approach**

**for Intrusion Detection System:**

An auto-encoder includes two parts: encoder and decoder.

Encoder aims to compress input data into a low-dimensional

representation, and decoder reconstruct input data based on

the low-dimension representation generated by the encoder.

On the other hand, auto-encoder can encode a representation

of an input layer into a hidden layer and then decode it into

an output layer [3].

For a given training dataset X = {x1, x2, ..., xm} with

m samples, where xi is a d-dimensional feature vector, the

encoder maps the input vector xi to a hidden representation

vector hi through a deterministic mapping fθ as given in (1)

hi = fθ(xi) = s(W xi + b), (1)

where W is a ´

d × d, ´

d is the number of hidden units, b is a

bias vector, θ is the mapping parameter set θ = {W, b}. s is

sigmoid activation function denoted as

s(t) = 1

1 + exp−t , (2)

where parameter t affects for the shape of the function.

The decoder maps back the resulting hidden representation

hi to a reconstructed d-dimensional vector yi in input space

as

yi = gθ

´(xi) = s(W h ´ i + ´b), (3)

where W´ is a d × ´

d, ´b is a bias vector and ´θ = {W , ´ ´b}.

The goal of training the autoencoder is to minimize the

difference between input and output. Therefore, a loss function

is calculated by the following equation

L(x, y) = 1

m

m

i=1

xi − yi

2 , (4)

where m is the total number of training dataset.

The main objective is to find the optimal parameters (θ and ´θ) which can be effectively minimize the difference between

input and reconstructed output over the whole training set as

θ = {W, b} = argθminL(x, y).

**Effective network intrusion detection via representation learning: A Denoising AutoEncoder approach:**

An autoencoder is employed for representation learning in this study to reduce dataset dimensionality and extract

meaningful features.

AutoEncoder is an

unsupervised learning technique, while the tabular model belongs to

the supervised learning category.

Autoencoder (AE) is an unsupervised-based neural network technique that represents the original data in a lower dimensionality and

then reconstructs it into its original dimension [30–32]. Its architecture is composed of an encoder and decoder. The encoder employs

a nonlinear transformation to map out the input data x to a hidden

representation z, while the decoder uses the nonlinear transformation

to reconstruct ̂𝑥 from z.

Vincent et al. [33] proposed denoising autoencoder (DAE), a

stochastic extension to the basic AE, designed to enhance the robustness of the original AE. DAE reconstructs the corrupted version of the

original data to its clean version. Through the stochastic mapping ̃𝑥 ∼

𝑞 (̃𝑥|𝑥), the original input data x is corrupted to a partially destroyed

input ̃𝑥. Next, the ̃𝑥 is propagated through DAE which are mapped

to the hidden representation z to finally be reconstructed as vector ̂𝑥.

Parameters of DAE are trained to reduce the total reconstruction error,

and it adopt an encoding procedure extremely effective in retrieving

the representative features to neutralize the noise effect’s and achieve

a better data reconstruction. DAE have attracted its application in lowdimensional data representation [34,35] and recover a noise-corrupted

input data [36].

**Deep Learning for Cyber Security Intrusion**

**Detection: Approaches, Datasets, and Comparative**

**Study:**

The deep auto-encoder was used by Shone et al. (2018)

[59] for cyber security intrusion detection. They use an autoencoder featuring non-symmetrical multiple hidden layers to

facilitate improved classification results compared with deep

belief networks. The study uses the KDD Cup ’99 and NSLKDD datasets with five metrics performances, including, accuracy, precision, recall, false alarm, and F-score. The results on

the KDD Cup ’99 dataset evaluation show that the proposed

model is able to offer an average accuracy of 97.85%, which

is better compared to the work in [45]. In addition, the results

on the NSL-KDD dataset evaluation show that the proposed

model offered a total accuracy rate of 85.42%, which is an

improvement upon the deep belief network model by 5%.

Khan et al. [60] proposed an intrusion detection system

based on the two-stage deep learning model, named TSDL.

The TSDL model uses a stacked auto-encoder with a soft-max

classifier, which is composed of three main layers, namely, 1)

the input layer, 2) the hidden layers, and 3) the output layer.

These three layers employ a feed-forward neural network

similar to a multi-layer perceptron. The study uses two public

datasets, including, KDD99 and UNSW-NB15 datasets. The

results on KDD99 dataset achieve high recognition rates, up

to 99.996%. In addition, the results on UNSW-NB15 dataset

achieve high recognition rates, up to 89.134%.

To design a self-adaptive and autonomous misuse intrusion

detection system, Papamartzivanos et al. [61] propose the use

of auto-encoder techniques. Specifically, the proposed system

is based in four phases, including 1) Monitor, 2) Analyze,

3) Plan, 4) Execute, and 5) Knowledge. The monitor phase

determines any alteration event that requires an intrusion

detection system adaptation. The analyze phase uses network

audit tools (e.g., Argus and CICFlowMeter) to perform the

transformation of the raw network traffic into network flows.

The plan phase uses a sparse autoencoder to learn representations of the input data. The execute phase is accountable

for storing purposes. However, the study uses two datasets

in performance evaluation, including KDDCup’99 and NSLKDD. The results show that the average accuracy of the static

model is 59.71% while for the adaptive model it is 77.99%

The combination of an improved conditional variational

autoencoder and deep neural network was used by Yang et

al. [62] for cyber security intrusion detection. The proposed

study consists of three phases: 1) training, 2) generating new

attacks and 3) detecting attacks. The training phase consists

of optimizing the loss of the encoder and the decoder. The

phase of generating new attacks uses a multivariate Gaussian

distribution as the distribution. The phase of detecting attacks

employs a deep neural network to detect attacks. To validate

the proposed model, the NSL-KDD and UNSW-NB15 datasets

are used, which the default learning rate of the Adam optimizer

is 0.001. The results show the highest accuracy of 89.08% and

detection rate of 95.68% on the UNSW-NB15 dataset.

To construct a deep neural network, Abusitta et al. [63]

uses a denoising autoencoder as a building block for cyber

security intrusion detection. The denoising autoencoder is used

to learning how to reconstruct intrusion detection systems

feedback from partial feedback. The KDD Cup 99 dataset is

used on the performance evaluation, which the results show

that the proposed model can achieve detection accuracy up

to 95%. Therefore, stacked denoising auto-encoders is used

by Wang et al. (2016) [64] for detecting malicious JavaScript

code. The study uses a dataset, which is composed of 12 320

benign and 14 783 malicious JavaScript samples. With three

layers of auto-encoders and 250 hidden units, the experimental

results show an optimal choice for building an intrusion

detection system based on the deep learning technique.

4) Deep auto encoders (DA): An autoencoder is composed

of both the encoder and the decoder, as presented in Figure 9.

Refer to [151], these two parts can be defined as follow:

encoderG (x) = s(W x + b) (10)

decoderG0 (y) = s(W0

y + b

0

) (11)

where G = {W, b}; G

0 = {W0

, b

0

}; W is a d

0 × d weight

matrix; x is an input vector; y is the hidden representation; b

is an offset vector of dimensionality d

0

.

**A deep learning approach for proactive multi-cloud cooperative**

**intrusion detection system**:  
3.2. The traditional autoencoders

An autoencoder is an unsupervised learning approach that is

used to learn efficient and reliable data codings [31]. It is used

to pre-train each layer in a deep neural network in order to

obtain better initial weights that lead to a better performing classification [8]. Researchers have seen that weights initialization

using autoencoders can improves the performance of deep neural

networks compared to random initialization [8].

The structure of an autoencoder is shown in Fig. 2. The dimensions for both input (IDSs’ feedback) and output (reconstruction

of IDSs’ feedback) are the same as shown in the figure. Labels

(i.e., +1) in Fig. 2 represent a bias vector. The use of biases in

a neural network increases the capacity of the network to solve

problems [32].

An autoencoder is used as a building block for deep networks [8]. In particular, it takes input vector (IDSs’ feedback)

x ∈ [0, 1]

d

, where d is the vector dimension, and maps it to a

hidden representation h ∈ [0, 1]

d

′

using the following equation:

h = fθ (x) = Sigmoid(W ∗ x + b) (1)

A. Abusitta, M. Bellaiche, M. Dagenais et al. / Future Generation Computer Systems 98 (2019) 308–318 311

Fig. 2. Example of an autoencoder.

θ = {W, b}, W is a weight matrix and b is a bias vector.

After that, the resulting hidden layer representation h will be

reconstructed to the output layer x

′ using a decoding function as

follows:

x

′ = gθ

′(h) = Sigmoid(W′

∗ h + b

′

) (2)

θ

′ = {W′

, b

′

}, W′

and b

′

are a weight matrix and a bias

vector of the reverse mapping, respectively. The weight matrix

W′ of the reverse mapping may optionally be constrained by

W′ = WT

, in which case the autoencoder is said to have tied

weights [8,10]. Each training x is thus mapped to a corresponding

h and a reconstruction x

′

.

The purpose of the model is to optimize the parameters (θ

and θ

′

) of the model, so that the reconstruction error between

input and output can be minimized. The following optimization

problem is used for this purpose:

θ

∗

, θ′∗ = arg minθ ,θ′

1

n

∑n

i=1

L(x

(i)

, x

′(i)

)

=

1

n

∑n

i=1

L(x

(i)

, gθ

′(fθ (x

(i)

)))

(3)

where L is a loss function. In an unsupervised learning model, a

loss function is a measure of how close the reconstructed input

is to the original input [32].

Since the input vector x is a binary vector (an IDS’s feedback

is either 0 or 1), a reconstruction cross-entropy is used as a

loss function [33]. Thus, the loss function will be described as

follows:

L(x, x

′

) = −∑

d

i=1

[xi

logx′

i + (1 − xi)log(1 − x

′

i

)]

The traditional autoencoder is then used to take corrupted

data z and try to learn how to reconstruct x. This is done by

allowing the input z to be mapped to a hidden representation as:

h = fθ (z) = Sigmoid(W′

∗ z + b

′

) (6)

Note that we selected z as input instead of x as a traditional

autoencoder was used. The result of the above equation h is then

used to reconstruct x

′

as follows:

x

′ = gθ

′(h) = Sigmoid(W ∗ h + b)

