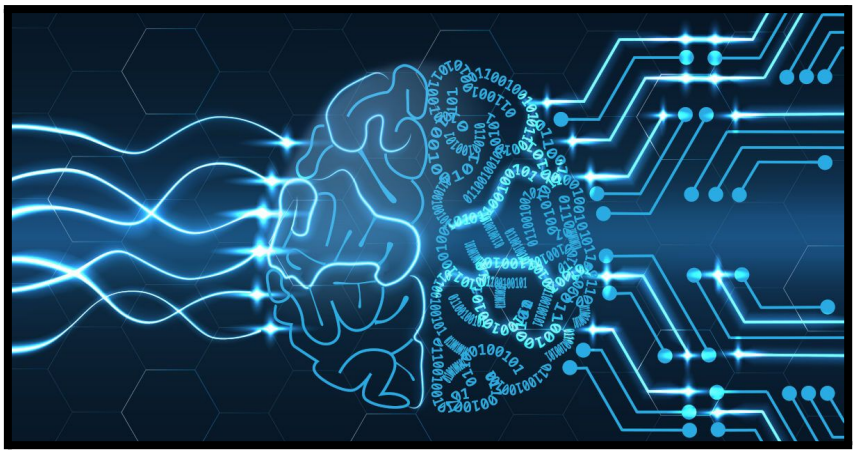
Ma513 - Hands on machine learning for cybersecurity

Chosen subject : n°2

Based on "'A Survey of Deep Learning Methods for Cyber Security"



Students:

Quentin Gourier | Morgan Grière | Léo Innocenzi

Teacher:

Leila Gharsalli

GitHub repository → *https://github.com/ileocho*
Survey article → *https://www.mdpi.com/article/ml_for_cyber*

Cyber / Data / IA - AERO 5
Institut Polytechnique des Sciences Avancées
63 Bd de Brandebourg Bis, 94200 Ivry-sur-Seine

Part 1

Explain the main idea of the article.

This article is an in-depth analysis of deep learning methods applied to cybersecurity topics. The authors are synthesizing a wide variety of deep learning applications to cybersecurity threats, from malware detection and classification to human behavior analysis, botnet detection and many other critical assets of today's information systems. The techniques analyzed in the article cover the vast majority of common deep learning algorithms and architectures, occasionally citing fascinating applications from other deep learning domains, like image processing and natural language processing.

To train deep learning models, the authors present a set of high-quality datasets with different applications, either from real data sources or machine-generated sources. Because this field is relatively recent, and because the type of data involved in detecting malwares or breaches in very important systems, they are often proprietary and not open-source, which is a massive drawback for researchers. These datasets are mainly focused on malwares but some include other types of data, like raw network logs or malicious domain names.

Deep learning is becoming increasingly important in the field of cybersecurity as it allows for the automation of threat detection and response. By analyzing patterns and behaviors within large sets of data, these algorithms can identify anomalies that may indicate a security breach or attack. Therefore, deep learning can also be used to predict and prevent future attacks by identifying potential vulnerabilities in a network or system.

One example of this is using algorithms for intrusion detection, where it can automatically detect and respond to potential network breaches. Overall, it can greatly enhance the ability to protect against cyber threats and improve incident response times. Once a model is trained, predictions can be made for real cases, it could radically change the evolution of cybersecurity for the next decades.

However, the article highlights the limits of the use of deep learning for cybersecurity starting with the increasing complexity and frequency of cyber attacks. Finally, the paper shows the need for more consistent benchmark datasets and future research that takes the empirical use of deep learning into account.

The trained models are true black boxes and can't be analyzed in specific cases. If, for any reason, an error comes out on a real-life application, it would be impossible to understand its nature because of their non-interpretability, a crucial drawback for a subject applying to cybersecurity.

Part 2

Discuss the presented methods, their advantages, and disadvantages.

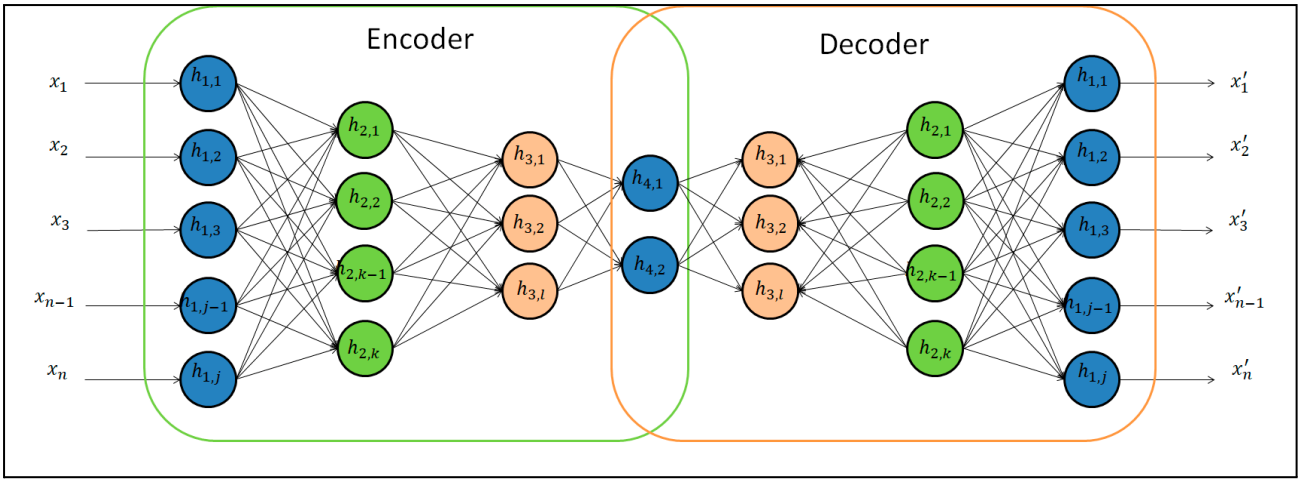
The article discusses various deep learning methods that are used in cyber security, including Deep Belief Networks (DBNs), Deep Autoencoders, Restricted Boltzmann Machines (RBMs), Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs) and Recursive Neural Networks (RecNNs). We will present their functioning, advantages and disadvantages.

Deep Belief Networks (DBNs) are composed of multiple layers of hidden units with connections between the layers, but not between units within each layer. They are trained in an unsupervised manner and are typically used for feature compression and removing noise. DBNs are particularly useful for image and speech recognition tasks, as they can extract features from raw data in an unsupervised manner. One of the main advantages of DBNs is that they can be trained one layer at a time, which reduces the computational resources required to build an effective model. However, one of the main disadvantages is that they can be sensitive to initialization and can be difficult to fine-tune.

Deep Autoencoders are a class of unsupervised neural networks that take as input a vector and try to match the output to that same vector.

The functioning of a deep autoencoder is based on the principle of encoding the input data into a lower-dimensional representation, also known as the bottleneck, and then decoding it back to the original dimension. The encoding and decoding are performed by two separate neural networks, the encoder and the decoder respectively, that are trained together. The encoder network takes the input data and reduces it to a lower-dimensional representation, while the decoder network takes this lower-dimensional representation and tries to reconstruct the original input. The objective of the training is to minimize the reconstruction error, which means that the output of the decoder should be as close as possible to the original input. This process allows the autoencoder to learn the most important features of the input data and remove the noise and redundancy.

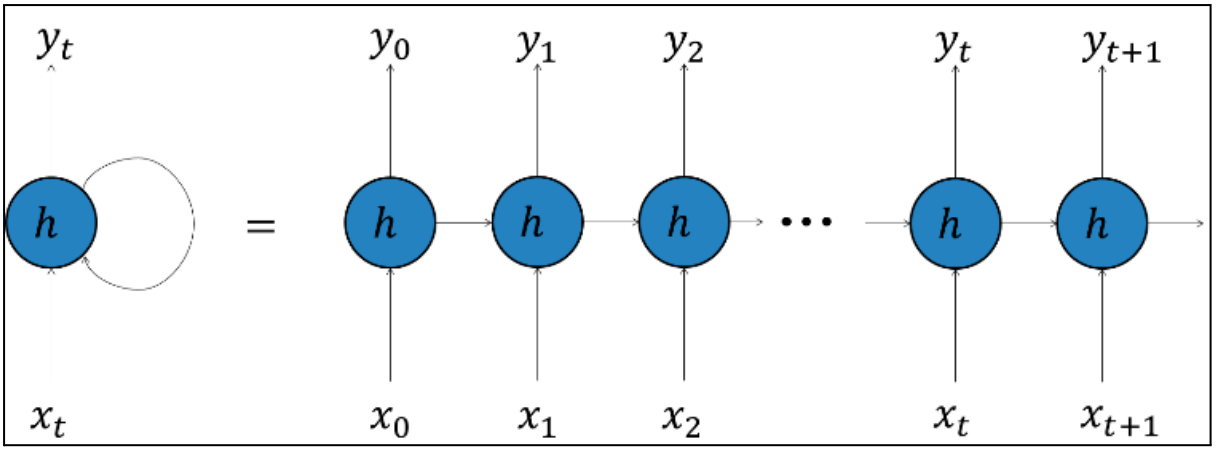
They are versatile and can learn compressed data encoding in an unsupervised manner and can be trained one layer at a time to reduce computational resources. Autoencoders have been used for a wide range of tasks such as image and speech recognition, anomaly detection and generative models. One of the main advantages of autoencoders is that they can be used for unsupervised feature learning and dimensionality reduction. However, one of the main disadvantages is that they can be sensitive to the choice of the architecture and the initialization.



Deep Autoencoder

Restricted Boltzmann Machines (RBMs) are a type of generative model that are composed of two layers of units: a visible layer that represents the input data and a hidden layer that represents the features of the input data. The connections between the layers are undirected, meaning that data can flow in both directions. RBMs are trained in an unsupervised manner, typically using a method called contrastive divergence. This method adjusts the weights of the network to minimize the difference between the input data and the output of the network. RBMs have been used for a wide range of tasks such as feature learning, collaborative filtering, and topic modeling. One of the main advantages of RBMs is that they can be trained efficiently using contrastive divergence. However, one of the main disadvantages is that they can be sensitive to the choice of the architecture and the initialization.

Recurrent Neural Networks (RNNs) are a type of neural network that are designed to handle sequential data. RNNs have a hidden state that is updated at each time step based on the input data and the previous hidden state. This allows RNNs to capture long-term dependencies in the data. RNNs have been used for a wide range of tasks such as language modeling, speech recognition, and machine translation. One of the main advantages of RNNs is that they can handle sequential data and capture long-term dependencies. However, one of the main disadvantages is that they can be difficult to train due to the risk of gradients vanishing or exploding. This problem is mitigated by using more advanced training techniques such as long short-term memory (LSTM) units and gated recurrent units (GRUs).

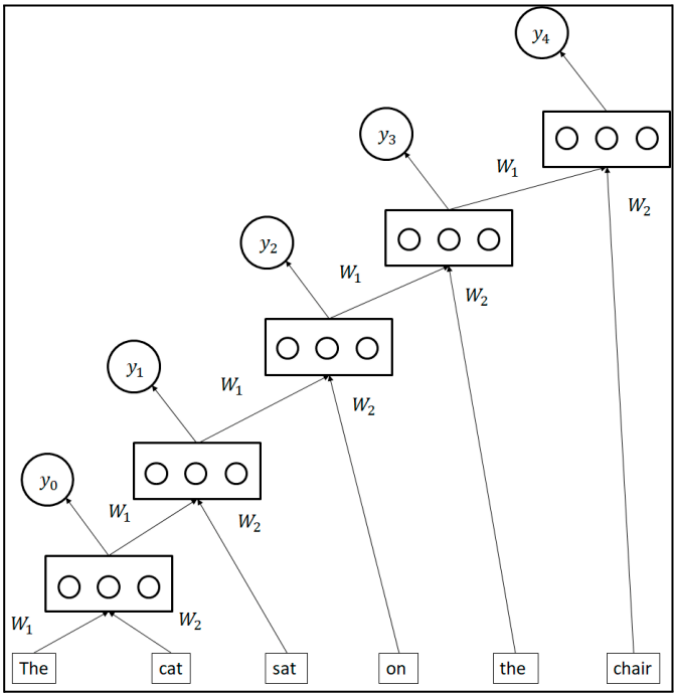


Recurrent Neural Network

Convolutional Neural Networks (CNNs) are used to process input stored in arrays such as images or videos. They are made up of convolution layers, pooling layers, and the classification layer. The convolution layers are the core of the CNN and use convolution kernels to analyze the input, while the pooling layers help to reduce memory and the classification layer is used to make predictions. CNNs have been used for a wide range of tasks such as image classification, object detection, and semantic segmentation. One of the main advantages of CNNs is that they can effectively learn features from images and other 2D data. However, one of the main disadvantages is that they can be computationally expensive to train and test.

Generative Adversarial Networks (GANs) are a type of network architecture used in unsupervised machine learning, in which two neural networks compete against each other in a zero-sum game to outsmart each other. Developed by Goodfellow et al. [46], one network acts as a generator and another network acts as a discriminator. The generator takes in input data and generates output data with the same characteristics as real data. The discriminator takes in real data and data from the generator and tries to distinguish whether the input is real or fake. When training has finished, the generator can generate new data that is not distinguishable from real data. GANs have been shown to be particularly useful for image generation and have been used in a wide range of applications such as image synthesis, image-to-image translation and image super-resolution. In terms of advantages, GANs are able to generate new data that is similar to the training data, allowing for the generation of new images, videos, and other types of data. Additionally, GANs are able to discover hidden features in the data that are not explicitly programmed into the model, allowing for more accurate and diverse results. However, GANs also have some disadvantages. Training GANs can be difficult and require a lot of computational resources. GANs can also be unstable and difficult to optimize, and it can be difficult to ensure that the generator and discriminator have converged to a stable equilibrium.

Recursive Neural Networks (RecNNs) are neural networks that apply a set of weights recursively to a series of inputs. They are used for various natural language processing tasks and image segmentation. The structure of RecNNs allows them to capture the hierarchical relationships present in the input data, making them particularly useful for tasks such as parsing and text classification. One of the main advantages of RecNNs is their ability to model hierarchical structures in the data. However, one of the main disadvantages is that they can be computationally expensive to train and test. Additionally, they can be sensitive to the choice of the architecture and the initialization.



Recursive Neural Network

In conclusion, the article presents several deep learning methods that are commonly used in cyber security, including Deep Belief Networks, Deep Autoencoders, Restricted Boltzmann Machines, Recurrent Neural Networks, Convolutional Neural Networks, Generative Adversarial Networks and Recursive Neural Networks. Each of these methods has its own strengths and weaknesses and is best suited for different types of tasks. GANs are a powerful method for image generation but are also difficult to train, and require a lot of computational resources.

Part 3

Use Deep Learning to predict cyber-attacks

In this part, we will apply deep learning methods to a cybersecurity topic. We were impressed by the application on malware mentioned in the article, hence our desire to focus our research in this direction. Before starting, we must give a definition of a malware.

In the context of engineering and cybersecurity, a malware is any malicious program or file that is designed to harm or exploit the functionality of a computer system. It is a type of cyber threat that is often used to steal sensitive information, disrupt operations, or gain unauthorized access to a network.

Our project is based on a feature of running malware : API calls. An API, or Application Programming Interface, is a set of programming instructions that allow different software applications to communicate with each other. Malware contained in a standard Windows application may use API calls to interact with the operating system or other software on a target computer in order to perform its malicious actions.

A sequence of API calls therefore can allow us to identify malware. It is close to a digital signature, a series of actions that makes it possible to know which program stands as the source of these requests.

The data used in this project comes from a Github repository. Malicious Windows applications were crawled from open sources, then hashed to get their respective malware application through VirusTotal, a malware analysis tool which contains one the biggest database of flagged malwares, giving us our labels. The actual application was processed by a sandbox environment called Cuckoo to log API call sequences from each application, giving us our training data.