Eleven Paths

WHITEPAPER

Artificial Intelligence: GANs and Autoencoders applied to CyberSecurity

30.05.2019

Telefónica CYBER SECURITY UNIT

elevenpaths.com

Contents

1. Introduction ... 3
2. Background ... 3
3. Concepts ... 4
- 3.1. What is Machine Learning? ... 4
- 3.2. What is a Machine Learning model? ... 4
- 3.3. What is a Machine Learning algorithm? ... 5
- 3.4. Supervised Learning Model ... 5
- 3.5. Unsupervised Learning Model ... 6
- 3.6. Reinforcement Learning ... 6
- 3.7. What is training in Machine Learning? ... 7
- 3.8. What is Deep Learning? ... 7
- 3.9. Neuron and Perceptron ... 8
- 3.10. What is a Neural Network? ... 9
- 3.11. Theory of Generative Models ... 10
- 3.12. Variational Autoencoders ... 10
- 3.13. Trick Reparameterization ... 14
- 3.14. GAN Training ... 16

4. Faceswapping with VAEs ... 17
5. Faceswapping with GANs ... 19
6. Audio Generation ... 20
7. Protection tools ... 22
8. Conclusions ... 25
9. References ... 26
About ElevenPaths ... 29
More information ... 29

Executive summary

Artificial intelligence and cybersecurity are two critical pillars of the digital transformation process that organizations and society in general are experiencing nowadays. The everyday life of this digital revolution has changed how humans communicate, interact, work and live; and this makes it necessary to become aware of those risks that new technologies bring along. All users may apprehend the advantages offered by artificial intelligence, but we must be careful and understand the risks it may entail as well. In this paper we will discuss about Machine Learning and Deep Learning by showing different approaches. The possibility that an attacker may impersonate identities, voices, images or videos through an AI poses a high risk for organizations and the whole society. Over this paper we will show artificial intelligence applications intended to protect users, as well as its most offensive face.

1. Introduction

The world as you know it is continuously changing. Digital revolution, digital transformation of companies and society itself is changing the way of living and interacting. Threats existing in the physical world tend to be digitized, so causing a huge increase in fraud.

Artificial intelligence applications and its evolution is already anchored in the everyday life of society and the future of companies. The use of AI for everyday tasks is already a reality, for instance, to access a purchasing platform that learns its clients' preferences in order to provide them with the best products.

Artificial Intelligence, Deep Learning and Machine Learning are current and even-changing terms. Even though making people's life easier is supposed to be one of their goals, they may be misused. One of the clearer examples is the proliferation of fake news, supported by videos generated by an AI that seem to be something they are not, since the AI can modify images or videos in order to introduce or delete some aspects. The AI is also capable of generating voices that fits perfectly with other people's voices.

These examples are without a doubt cases related to the cybersecurity and awareness that must be implemented in the new society: the digital transformation society.

2. Background

Concerning the background, a number of examples on artificial intelligence uses to perform identity attacks or fraud against other users may be found. One of the more well-known examples is a video of Barack Obama where the ex-president appears apparently giving a message wanted by someone else.

This video was posted on 19 July 2017 [1] and was generated by using a model trained to create the mouth and its movement with the words desired. In particular, the text was mapped to Obama's unreal mouth. All this was generated by an AI after a long training. Here we can see the turning point between risk and newness. It is not a set-up made by someone, but the video is created by an AI that makes a swapping of Obama's mouth and generates the same mouth movement that when pronouncing such text. Moreover, the voice may be generated by an AI as well; that is, an AI may be trained to generate a voice identical to Obama's one.

This issue opens a debate: Is what is watched and listened true? Can society rely on what they watch and listen? This constitutes one more step in the fake news world, since it opens up a wide range of possibilities. It can be seen as an evolution of phishing and social engineering techniques that could lead anyone to believe that they are truly watching the person concerned and this is really giving the message.

Other examples of Faceswapping possibilities –regardless of the technique or technology used– are the Jennifer Lawrence and Steve Buscemi ones [2]. By mentioning this example, the intention is not to show fraud, but the AI capabilities to make a swapping by placing someone's face on someone else's body. This video has been generated by Autoencoders and shows the features of the source person and the face of the target person. After a huge training, the AI learns to generate frames with Jennifer Lawrence's body, her features and expressions, but with Steve Buscemi's face.

3. Concepts

In this section we will go over some essential concepts related to machine learning that must be clear to understand the functioning of technologies involved in generating this type of contents as well as in creating resources to protect us against such contents.

3.1. What is Machine Learning?

Machine Learning may be defined as "a machine's capacity to learn from experience". The aim of Machine Learning is to create algorithms that may learn how to perform a particular task on the basis of sample data.

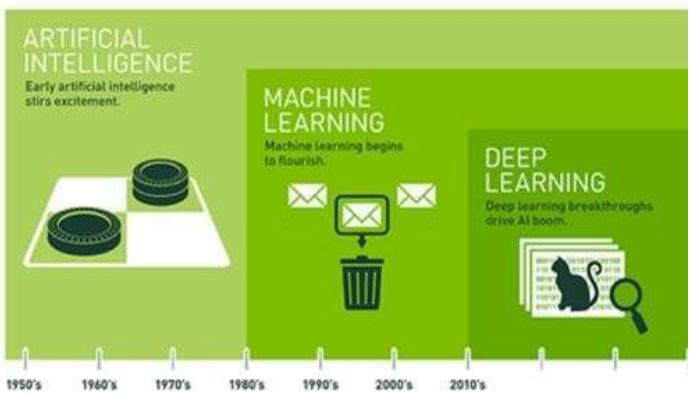

Fig. 1. Relationship between Artificial Intelligence, Machine Learning and Deep Learning [3]

Machine Learning is a branch of Artificial Intelligence whose goal is to extract as many information and knowledge as possible from data. In recent years, Machine Learning has become omnipresent in our lives, since it is present in multiple services that we use every day.

Machine Learning most basic use is the practice of using algorithms to parsing data, learn from them and then be capable to make a prediction or suggestion about something. The machine is trained by using a great amount of data, so making it possible for the algorithms to be improved.

3.2. What is a Machine Learning model?

A machine learning model is a mathematical representation of a real-world process. To generate a Machine Learning model, it is necessary to provide training data to a machine learning algorithm, so that this algorithm may "learn". This way, a model is simply a function capable of making a prediction.

3.3. What is a Machine Learning algorithm?

ML algorithms are those that can learn from data and improve from experience, without human intervention. Learning tasks may include learning the function that assigns the input to the output or learning the structure hidden within unlabeled data. Formally, an algorithm is a set of steps performed in a given order.

A Machine Learning model, such as regression, clustering or neural networks, are based on the functioning of algorithms for them to be executed. The algorithms are the engine that underlies the code to be executed. They perform the hard work of multiplying matrices, optimizing results and generating an output. There are many kinds of models, including a complete machine learning ecosystem in a wider context.

The word "model" is quite nebulous and difficult to be separated from something as a "function" or an "equation". Given that separating the concepts of model and algorithm seems to be a difficult task, a ML algorithm may be defined as a set of steps that go through a model for this one to make the calculations and perform processing.

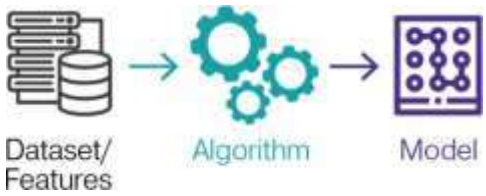

Fig. 2. Pipeline of training a Machine Learning model [4]

3.4.Supervised Learning Model

The most used algorithms in Machine Learning are those capable of automating a process after having learnt from a set of known examples. All the algorithms covered by this premise are grouped as supervised learning algorithms [5], on which the user provides a given algorithm [6] with pairs of input-output examples, so the algorithm must learn the relation in order to later be capable of providing, without human supervision, an output for an input never seen.

For supervised learning problems, the algorithm is taught or trained from data already labeled with the correct answer. The larger the data set, the better the algorithm may learn on the matter. When the training is concluded, new data is provided, but this time without labels of correct answers, so the learning algorithm uses past experience acquired over the training phase to predict a result.

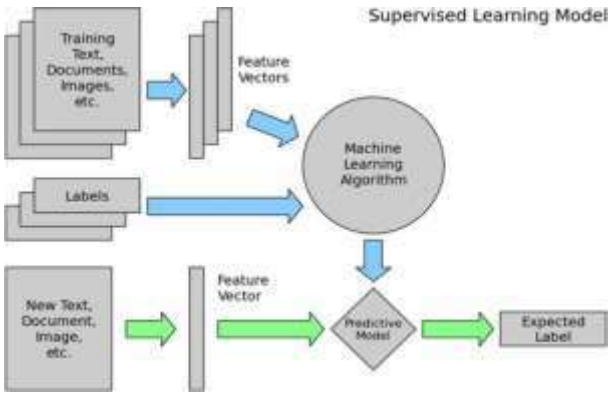

Fig. 3. Diagram of training and testing under supervised model [7]

There are two main methods of supervised learning, depending on the type of ML tasks that they are intended to perform: classification and regression. We can illustrate it using the example of a spam detector. The model on which that filter is based will be fed with a set of malicious mails labeled as illegitimate, together with another set of non-malicious mails identified as legitimate. Once the model is trained, it will be capable of discerning if a mail (not involved in the model training) is legitimate or not.

3.5. Unsupervised Learning Model

The other model corresponds to unsupervised learning, where the model is fed only with input data. There are multiple applications of this type of method, even though they may be harder to evaluate. On unsupervised learning problems, the algorithm is trained by using a set of unlabeled data. In this case, the algorithm is not told what the data represents. The algorithm is intended to find, on its own, patterns that could help it to understand the whole set of data. Unsupervised learning is similar to the method used to teach children to speak, since at first they listen to people speak around them but without understanding anything. Over time, when they have listened tens of conversations, their brain will start to build a model on how language works, and this will allow them to identify a number of patterns and act according to certain sounds.

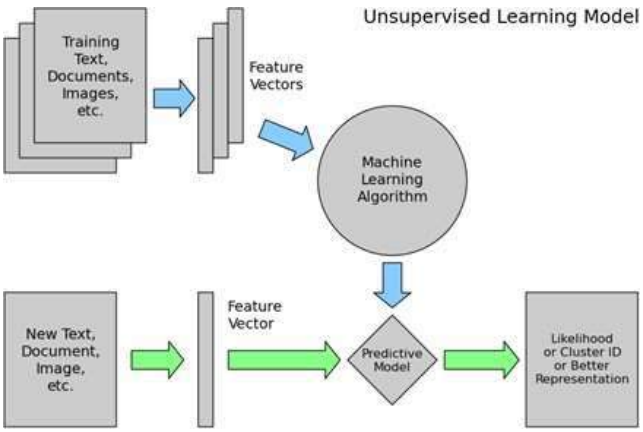

Fig. 4. Diagram of training and testing under unsupervised model [7]

Some of the most common algorithms on unsupervised learning are:

- Density estimation: based on finding the probability function of an independent variable given a set of observations defined by such variables.
- Clustering: mainly aimed to find the set of uncategorized inputs with common features.
- Feature Learning: set of techniques that allow a system to find the necessary representations to detect or classify features from unprocessed data.
- Dimension reduction: aimed to reduce the number of random variables of a sample in order to define it under a set of key variables. Some of the most used ones are Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE).

3.6. Reinforcement Learning

Among these unsupervised methods, we can include Reinforcement Learning as well. On Reinforcement Learning problems, the algorithm learns by observing the world around. Its input data is the feedback or reward when facing a

successful observation, as an action response within a particular environment. In this case, the system learns by trial and error through an exhaustive investigation on the consequences of its actions within the environment.

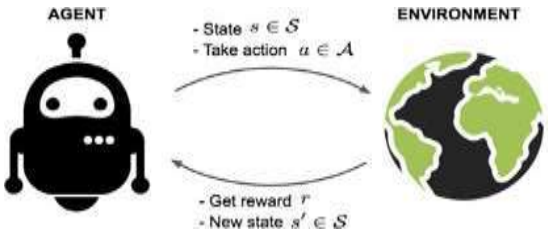

Fig. 5. Basic learning model of an agent through reinforcement learning [8]

The main steps of a reinforcement learning method are:

1. Preparing an agent with a set of initial policies and strategy.
2. Observing the environment and current status.
3. Selecting optimal policies and performing an action.
4. Obtaining the relevant reward (or punishment).
5. Updating the policies when necessary.
6. Repeating iteratively steps 2-5 until the agent learns the optimal policies.

3.7. What is training in Machine Learning?

The training process of a ML model involves providing a ML algorithm (that is, the learning algorithm) with training data so that this may learn, so adapting us to the initial conditions of data layout imposed by each paradigm.

3.8. What is Deep Learning?

Following the development of Machine Learning, over the last decade a Machine Learning particular technique known as Deep Learning has spread more strongly.

Over last years, Deep Learning has been developed, promoted by the use of devices such as Graphics Processing Units, and the most recent Tensor Processing Units, that have allowed researches on the improvement of faster architectures.

Fig. 6. Cloud TPU v2 Pod Alfa 11.5 petaflops 4 TB of HBM. 2D Toroidal Mesh Network. [9]

The brain is the most fascinating organ of the human body. Thanks to this organ we are able to process absolutely everything around us through sight, hearing, smell, taste and touch. But not only that, our brains also allow us to store memories, experience emotions, and even dream. Without the brain we would be primitive organisms unable to perform any simple action. The brain is unfailingly what makes us intelligent beings.

Child's brain only weighs 0.5 kg, but somehow it can solve complex problems that are practically impossible to deal with by powerful computers. In a matter of months, those little brains can recognize faces and voices as well as identify

objects over long distances. They also start to associate sounds with specific meanings. In early childhood, the brain incorporates gradually and exponentially new capabilities, so acquiring a sophisticated comprehension of grammar and including thousands of words in child's vocabularies.

Over decades, we have dreamed of developing intelligent machines with brains similar to ours: robot assistants to clean our homes, autonomous driving cars, systems capable of detecting diseases at a hit rate above humans' one... and we are achieving it. However, developing and building these machines involve facing new issues that force us to solve computational problems to perform tasks that our brains may achieve in less than a few seconds. To address some type of problems, we have been forced to wait for new technologies, to develop a new and radically different programming method, as well as to develop new algorithms that may speed up the accomplishment of some tasks.

Why computers find certain problems so difficult to solve? Traditional computers are designed to be very good at two aspects: 1) performing quick arithmetical calculations and 2) performing those calculations by following an explicit list of instructions.

By definition, Deep Learning is a subgroup within the field of Machine Learning. In Deep Learning, rather than teaching computers an endless list of rules to fix a problem, we provide it with a model to evaluate examples and a little collection of instructions to modify the model when errors occur. Over time, we expect that these models will be able to extremely precisely solve the problem, since the system is capable of extracting patterns. Although there are various techniques for implementing Deep Learning, one of the most common ones is simulating an artificial neural network system within the data analysis software.

Neural Networks are a computational model that shares some properties of animal brain, where many simple units undertake parallel work without a centralized control unit. Weights between units are the main long-term information storage medium in Neural Networks. The main way for a neural network to learn is by updating weightings. The behavior of a neural network is defined by the architecture of its network. The architecture of a neural network is partly characterized by the following elements:

- The number of neurons
- Their depth or number of layers
- The connection existing between layers

3.9. Neuron and Perceptron

Neurons can be understood as the basic units of the human brain. A minor part of the brain, about the size of a grain of rice, contains more than 10,000 neurons; each one with an average of 6,000 connections with other neurons. Thanks to this massive biological network we are able to experience the world around us. Our goal in this section will be emulate such structure in order to use it for building a machine learning model intended to solve problems in a similar way.

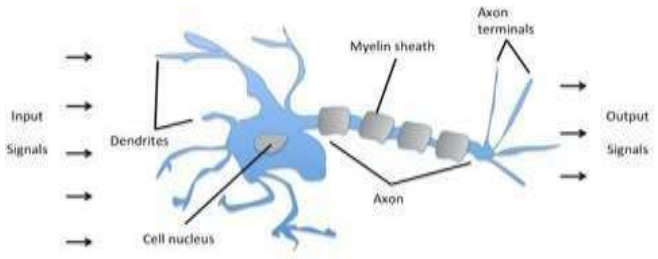

Fig. 7. Schematic of a Biological Neuron. [10]

At its nucleus, a neuron is optimized to receive information from other neurons, process such information in a unique way and send its result to other nerve cells. The process is summarized in the Figure above. The neuron receives its inputs via structures similar to feelers, called dendrites. Each one of these input signals is dynamically strengthened or weakened, depending on the frequency they are used. The strength of each connection determines the neuron input until it outputs. After being weighed by the strength of their respective connections, inputs are added together in the cell body. This addition becomes then a new signal that is spread along the cell axon and later sent to other neurons.

Individual biological neurons seem to have a simple behavior, but they are organized in a large network of billions of neurons, each one typically connected to other thousands of neurons. This large but quite simple network can perform highly complex calculations. The brain altogether works this way, and what is more important: in a decentralized way where all parts are connected while performing parallel calculations.

Warren McCulloch and Walter Pitts (1943) proposed a simple model of biological neuron, later known as artificial neuron: it has one or more binary gates (on/off) and a binary output. McCulloch and Pitts showed that even with such a simple model it was possible to build a network of artificial neurons capable of performing any calculation on logical propositions. At this point is where the concept of perceptron was born. A perceptron is a binary classifier with a simple input-output relationship similar to the one existing in a dot product of *n* number of inputs with their associated weights, and later this "network input" is sent to a step function with a defined threshold. This step or activation function is usually a Heaviside step function with a given threshold value. This function will trigger a single real-value binary depending on the input.

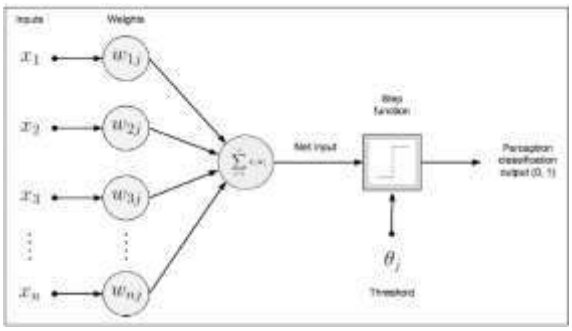

Fig. 8. Single-layer perceptron [11]

3.10. What is a Neural Network?

The best known and easy to understand neural network is the feedforward multilayer neural network, that performs a perceptron stacking. It has an input layer, one or several hidden layers and a single output layer. Each layer may have a different number of neurons and is completely connected to its adjacent layer. Connections between neurons within layers create an acyclic graph, as it can be seen in the figure below.

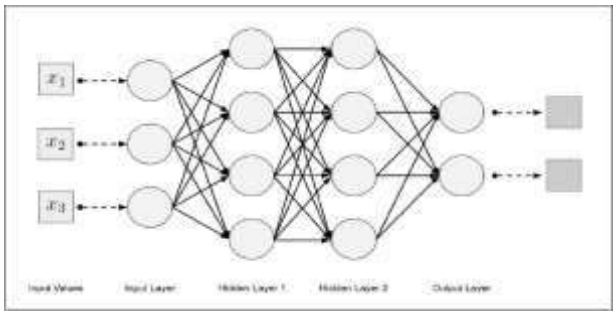

Fig. 9. Multi-layer neural network topology [11]

A multi-layer neural network may represent any function provided that it has enough units of artificial neurons. In general, it is trained through a learning algorithm.

Learning algorithms (associated to concrete network models) allow to modify weights of synaptic connections for the network to learn from those examples it faces.

Regarding Deep Learning, the aim is to provide neuron layers with enough data to identify patterns, classify and categorize them. One of the major advantages is that work is carried out based on unlabeled data and that behavior and occurrence patterns are analyzed.

For instance, you can take an imagen as input information of the first layer. There it will be partitioned into thousands of parts that each neuron will separately analyze. We can analyze the color, form, etc. Each layer is specialized in a feature and assigns it a weight. Finally, neurons that make up the ending layer collect that information and provide a result.

3.11.Theory of Generative Models

A Generative Model is aimed to learn any type of data distribution through unsupervised learning. All types of generative models are intended to learn the real data distribution of the training set and, when done, be capable of triggering new samples with slight variations. It is not always possible to abstract the exact distribution of our data, so by using Neural Networks we try to shape a distribution as similar as possible to the real one.

While a discriminative model is a conditional probability model of a target given an independent variable X, generative models allow to calculate the joint probability distribution between an observation and a variable.

In recent years two deep architectures have become the main pillars to develop generative models: Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). VAEs try to maximize similarity between generated images and those images used to feed data logs; GANs are trained with the aim of achieving a balance between a generator and a discriminator.

3.12.Variational Autoencoders

In contrast to the most standard uses of Neural Networks as models to perform regression or classification, VAEs are powerful generative models that in recent years are meeting applications in very diverse fields, ranging from human face creation to synthetic audio production.

Neural Networks may acquire countless possible configurations regarding their form and size. Indeed, these form and size determine the architecture performance to fix a given problem.

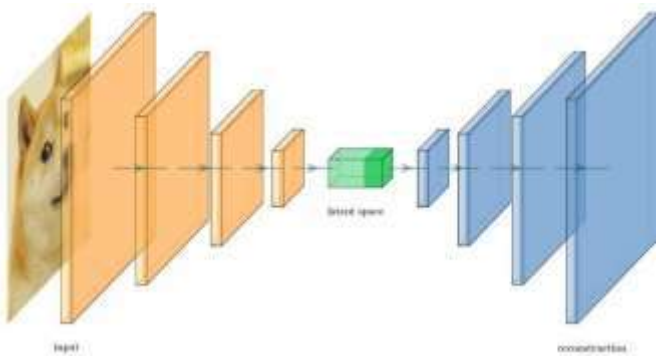

Fig. 10. Autoencoder typical architecture [12]

An autoencoder is a type of artificial neural network aimed to generate a representation as close as possible to its original input. At first sight, it does not seem to fix any real problem, but the value of this type of architectures relies on the features that may be extracted from input abstraction performed over the deep layers of its architecture. The autoencoder must compress the information to reconstruct it afterwards.

After an accurate training, the autoencoder will learn how to represent the input values in a different but much more compressed manner. Generally, in a conventional neural network, one tries to predict a target vector on the basis of input vectors. It is essential to learn an assignation for the network not to have restrictions. However, if the network is restricted the learning process becomes more interesting.

The simplest Autoencoder (AE) has an MLP (Multi-Layer Perceptron) structure. Autoencoders do not require any label, they may be trained under an unsupervised learning model.

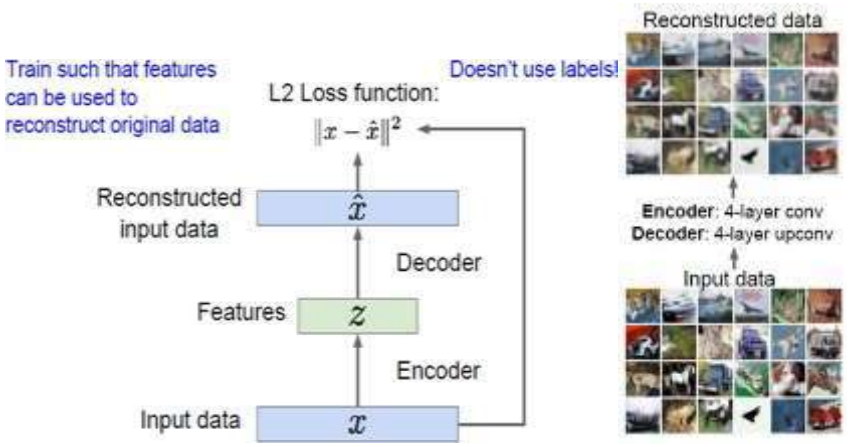

Fig. 11. Architecture and functioning of an Autoencoder [13]

The only restriction of this type of architectures is that the dimension of hidden layers is always smaller than the size of the input one. This way, the amount of information that flows through a neural network is limited, so forcing the autoencoder to learn only those important attributes of input data that allow it to reconstruct the input status from a codified status. Ideally, over image compression the model will learn to extract the latent attributes of input data.

The diagram above shows an image that feeds an encoder. Its result is a dimensional underrepresentation of that image, sometimes called base vector or latent image. Depending on the network architecture, latent face might not seem a face at all. When it goes through a decoder, the latent face is rebuilt. Autoencoders leak, so it is highly unlikely that the rebuilt face has the same detail level originally existing.

An *Autoencoder* may be mathematically defined as:

- An input *x*.
- An activation function applied to each neuron within hidden layers *a(h)*.
- $W_i \in R^{I \times O}$ as the weighing matrix of the i-th dimension layer, where I is the input size and O the output size.

- $b_i \in R^O$ as the bias matrix of each layer

$$z = a\ xW_1 + b_1$$

$$x' - a(zW_2 + b_2)$$

So, the simplest Autoencoder may be summarized as follows:

x x'

Another loss function that is tried to be minimized when using Autoencoders is the cross-entropy function:

$$L\ = E\ x, x'\ -\ |x - x'|\ _{\rho}$$

$$L\ x, x'\ = -\sum_{j=1}^{M} x'_j\ log\ log\ x_j$$

Where M is the input data x dimensionality, which is associated for example to the number of pixels on an image.

However, this type of simple networks cannot generate invariable images beyond the typical resolution loss of Autoencoders. In order to substantially modify the output image towards a suitable direction, our model must be capable of learning the probability distribution of the input sample. Here is when Variational Autoencoders appear.

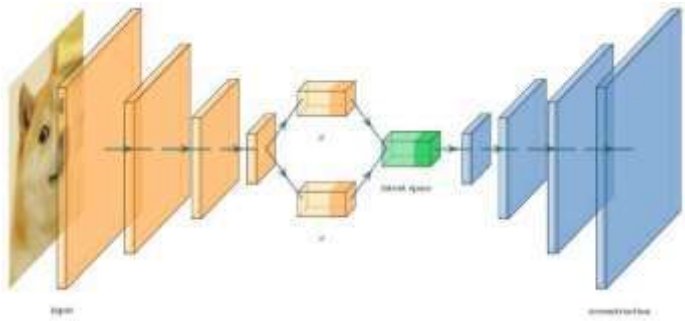

Fig. 12. Typical architecture of a Variational Autoencoder [12]

This is one of the most popular approaches to learn a complex data distribution, as it may happen with images unsupervisedly processed by Neural Networks. It is a probability model rooted in Bayesian inference. That is, the model is aimed to learn the underlying probability distribution of the training data, so that it can easily sample new data from that learned distribution. The idea is the same as for Autoencoders: learning a latent and low-dimensionality representation of learning data, called latent variables -variables not directly observed, but inferred from a mathematical model-, that are assumed to have generated our real training data.

In this case, the aim is modeling an input to our neural network as a distribution. Let's call this distribution $\frac{p_o}{x}$. The relationship between the input data x and the latent vector may be defined as follows:

- Prior $p_\theta\ z$
- Likelihood $p_\theta\ x|z$
- Posterior $p_\theta\ z|x$

If we assume that the parameter for this distribution is known, in order to generate a sample similar to a real network input, the following steps must be undertaken:

1. Firstly, a sample of $z^{(i)}$ from a prior distribution $p_{\theta^*}(z)$ is obtained.
2. By doing so, we can generate a value x^i from a conditional distribution $p_{\theta^*}(x \mid z = z^i)$

The optimal parameter is the one that maximizes the probability of generating samples identical to real data. That is:

$$\theta^* = \arg\max\left(\prod_i p_\theta(x^i)\right) = \arg\max\left(\sum_i \log p_\theta(x^i)\right)$$

The equation is updated to better show the data generation process to involve the encoding vector:

$$p_\theta(x^i) = \int p_\theta(x^i \mid z)\, p_\theta(z)\, dz$$

Unfortunately, it is not easy to define $p_\theta(x^i \mid z)$ to all potential values of *z*. In order to make a faster search, a new function approximation may be entered to output, given an input *x*, *q (z | x)*, parameterized by ø.

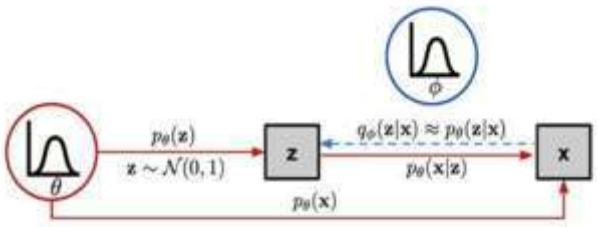

Fig. 13. Variational Autoencoder sampling [14]

Now we have a structure which is similar to an Autoencoder:

- The conditional probability *pø (x|z)* defines a generative model, similar to the decoder. This probability is also known as probabilistic decoder.
- The function approximation *qø (z|x)* is known as probabilistic encoder.

The estimated posterior *qø (z|x)* must be very close to the real *pø (x|z)*. We may use the Kullback-Leibler divergence [15] to measure the difference between these two distributions. KL *DKL (X|Y)* divergence measures the amount of information lost when distribution *Y* is used to represent *X*. The aim is to minimize DKL (*qø (z|x)|| pø (z|x))* (reversed KL) regarding ϕ. The following article on Variational Bayesian Methods [https://blog.evjang.com/2016/08/variational-bayes.html] explains why minimizing this way is chosen over DKL (*qø (z|x)|| pø (z|x))* (*forward* KL).

If KL divergence is developed:

$$D_{KL}\left(q_\phi(z\,|\,x)||p_\theta(z|x)\right)$$

$$= \int q_\phi\ z|x\ \log\frac{q_\phi\ z|x}{p_\theta\ z|x}dz$$

$$= \log p_\theta\ x\ + D_{KL}\left(q_\phi\ z|x\ \|\ p_\theta\ z\ \right) - \mathbb{E}_{z\sim q_\phi}\ z\,x\ \log p_\theta\ x|z$$

So updating the equation:

$$\log p_\theta\ x\ -\ D_{KL}(q_\phi\ z|x\ |\ p_\theta\ z|x\ ) = \mathbb{E}_{z\sim q_\phi}\ z|x\ \log p_\theta\ x|z\ - D_{KL}\left(q_\phi\ z|x\ \|\ p_\theta\ z\ \right)$$

The left-side term of the equation is exactly what is wanted to be maximized over the real distribution learning of network inputs: we wish to maximize the *log p0 x* probability to generate real data and minimize the difference between the actual and estimated DKL (*qø (z|x)|| pø (z|x)).*

The loss function of Variational Autoencoder may be defined as follows:

$$L_{\text{VAE}}\ \theta,\phi\ = -\log p_\theta\ x\ + D_{KL}(q_\phi\ z|x\ |\ p_\theta\ z|x\ ) = -\mathbb{E}_{z\sim q_\phi}\ z\,x\ \log p_\theta\ x|z\ + D_{KL}\left(q_\phi\ z|x\ \|\ p_\theta\ z\ \right)$$

$$\theta^*,\phi^* = \arg\min_{\theta,\phi} L_{\text{VAE}}$$

On Variational Bayesian Methods, this loss function is known as variational limit inferior. It is called "limit inferior" because KL divergence is always positive and therefore, *-L VAE* is the limit inferior of *log p0 x.* For this reason, by minimizing the loss function, the limit inferior of the probability that real samples may be triggered is maximized.

3.13. Trick Reparameterization

We need to trigger samples of *z ~ qø (z|xi)* in order to obtain new output data. Nevertheless, we cannot spread back the gradient during the training if we are stochastically triggering samples of such distribution. We need a resource to help us. In the case of VAEs, it is necessary to sample from a Gaussian distribution within the latent space of the network. The trick is to say that the sampling of *z ~ qø (z|xi) ~ N (z; ui, o2i I)* is equivalent to the sampling of *e ~ N (o, I)* and defining *z = u + o · e*, in reference to the product by elements of two matrices.

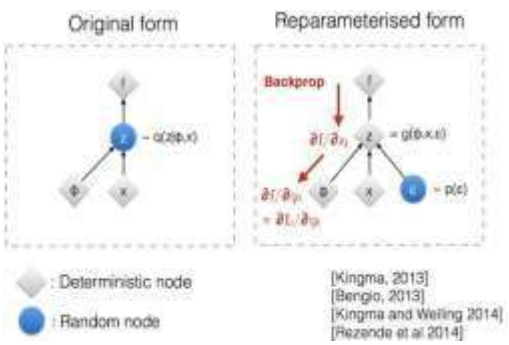

Fig. 14. Trick Reparameterization for Variational Autoencoders [16]

In the current case, we proceed with the training of the *u* and distribution model, which allows us to randomly sample the distribution *e ~ N (o, I).*

GANs are the perfect example of a neural network that constitutes a generative model by using the unsupervised learning model to train two models in parallel in a zero-sum game. A key aspect of GANs (and generative models in general) is how they use a given parameter count which is significantly lower than usual regarding the amount of data with whom the network is being trained. Consequently, the network is forced to properly represent the training data, making a more efficient data triggering easier, since these data must be similar to such set of training data.

The idea behind the learning of this type of architectures is quite simple but clever. Two models are simultaneously trained: a generative model G obtains the data distribution and triggers samples from a statistical model, while a discriminative model D -as a type of classifier- estimates the probability that a sample may come from the data used to train the model or that it may have been generated by G. The training of this type of networks seek to train D in order to detect incorrect generations by G, so maximizing the probability that D is wrong.

When the discriminative model D, that works as a binary classifier, rejects an example triggered by the generator, the generative model learns to increasingly better sharpen the generation of new samples. How is generator G able to trigger examples increasingly closer to reality? At every step of the training, D discriminator reports the generator how close it has been to a real example.

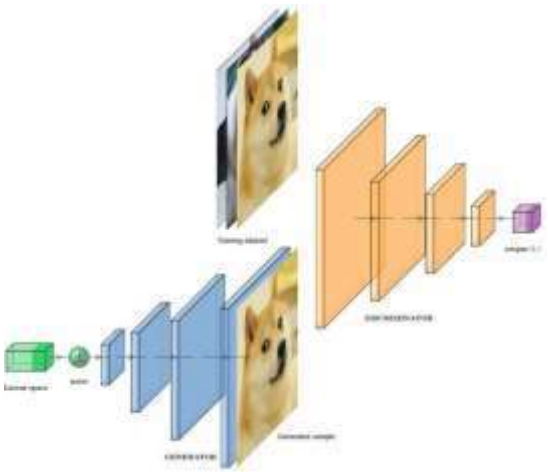

Fig. 16. Typical architecture of a GAN [12]

For example, when modeling images, the discriminative model is usually a standard CNN. The use of a secondary neural network as the discriminative one allows GAN to train both networks in parallel in an unsupervised way.

These discriminative networks take images as an input and then issue a classification. The gradient output of the discriminative network regarding input synthetic data will show to the generative network what minor changes must be performed in order to trigger more realistic synthetic data. The generative part of the GANs are usually of "deconvolutional" nature.

Over the training, the algorithm *backpropagation* is used on both networks with the aim of updating the weights and biases of the generator, so that more realistic output images can be generated. Through the training, it is sought to update generative network parameters to the extent that the discriminative model is "misled" by the realism of the images generated in comparison to the training data, so the model is unable to correctly classify the image as real or false.

For each iteration, a set of real and false images is taken, and the weighs of the generative and discriminative network are updated through *backpropagation*. According to *Goodfellow* et al. *Generative Adversarial Nets* [18], the problem lies in how difficult is to optimize this type of architectures, since a saddle point is tried to be found—instead of a minimum of the function— so the optimization process is unstable.

3.14. GAN Training

If we have a real image *x* and we define the random noise generated as *z*:

- *D (x)* is defined as the discriminator output when it is provided with a real image.
- *G (z)* is an image obtained from the generator.
- *D (G(z))* is the discriminator output to an image obtained from the generator.

Ideally, our discriminator would have the following behavior:

1. The discriminator should return a 1 as long as it has been provided with a real image, so *D(x)* should be maximized;

2. The same discriminator must assign a 0 to every image *G (z)*, so *D (G(z))* should be minimized. Therefore, *1 – D(G(z))* should be maximized. Consequently, we want to maximize 1 and 2, so our target function is:

$$\max_D V(D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D\ x\ ] + \mathbb{E}_{z \sim p_z(z)}[\log\left(1 - D(G\ z\ )\right)\ ]$$

For the generator G, the target function must be matched by image generation with the maximum *D(x)* value—that is, we must minimize V. We intend the generator to mislead the discriminator. Therefore:

$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)}[\log\left(1 - D(G\ z\ )\right)\ ]$$

Constructing and training a generative network with two linked architectures is a challenge that we had not faced until this type appeared: we have two loss functions, the generator's one, intended to create better images, and the discriminator's one, focused on distinguishing generated images from real ones.

The training of both models is simultaneous, so as the discriminator improves real images compared to the generated ones, the generator can better align its weighs and biases in order to generate plausible images. In this case we are facing a discriminator and a generator engaging a zero-sum game, where profits or losses of one are balanced with losses or profits of the other.

If we combine both target functions:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D\ x\ ] + \mathbb{E}_{z \sim p_z(z)}[\log\left(1 - D(G\ z\ )\right)\ ]$$

Once loss functions have been defined, optimizers remain to be defined. The optimizer is independent from each network, and we seek to keep fixed the weighs of a network while the weighs of the other network are being updated.

The optimizer will minimize the loss function for both the two loss functions of the discriminator and the generator's one.

It may be difficult to achieve GAN convergence, and generally it requires long time of training. This is not the only problem: we must also face the difficulty to choose the optimal architecture together with the appropriate hyperparameters. The fact of not choosing the appropriate configuration may lead us to fail in network convergence.

As we mentioned before, for GANs there is no guarantee of a single solution that *pdata (x) = pg*. We may find the known as "Failure to Improve". There are two ways for our generator to stop improving, where improvements are taken regarding the quality of the samples produced. In the first case, although there is a solution, the dynamics of Gradient Descent training algorithm avoid that Neural Networks reach their optimal parameter values.

The remaining case corresponds to the gradient along which the generator should be trained, it is reduced to the extent that the generator cannot learn from it. This is known as "vanishing gradient problem" or "saturation problem". According to Goodfellow et al. (2014), this problem is caused by the discriminator, since this is able to highly effectively reject generator samples, so the generator cannot keep learning. This is why discriminator overtraining must be avoided.

Another well-known training error is Mode Collapse. It is a problem occurred when the generator learns to trigger just a limited range of real data distribution samples. It makes it by assigning several and different input values to the same output point. This breakdown mode can be solved by adjusting the architecture or facilitating the generator how close it has been to mislead the discriminator. This way, the training is more efficiently managed.

4. Faceswapping with VAEs

In previous sections we have discussed the concept of Autoencoder and how Neural Network can use it to compress and decompress images.

Training a neural network means optimizing its weighs with the aim of achieving a specific objective. In the case of a traditional Autoencoder, network performance is measured according to how the original image is reconstructed from its representation on the latent space. It is important to point out that if we train separately two Autoencoders, they will be mutually incompatible. Latent faces are based on specific features that each network has considered significative over their relevant training process.

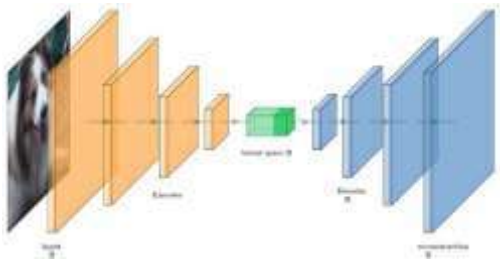

Fig. 17. Two Autoencoders sharing the same decoder learning various distributions [12]

What makes it possible face-swapping technology by using Autoencoders is to find the means for forcing both latent faces to encode with the same features. This may be achieved by making both networks to share the same encoder but using two different decoders. By doing so, once the neural network training has been completed, the following action may be performed over the testing of our model:

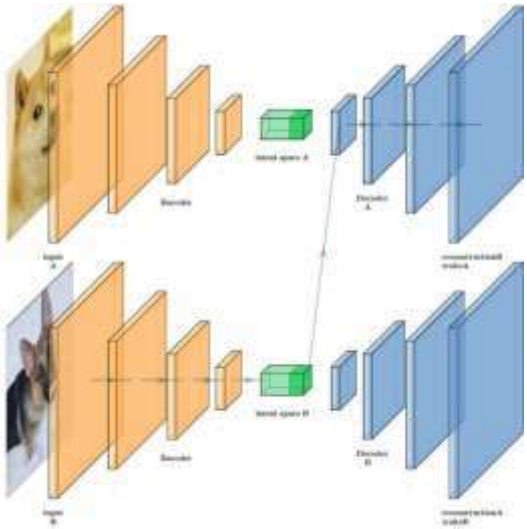

Fig. 18. Testing a VAE network focused on Faceswapping [12]

If the network becomes widespread enough to make a face, the latent space will represent facial expressions and orientations. This means generating a face for Subject A with the same expression and orientation as Subject B and the vice versa. The important point is that the two themes used for the training share as many similarities as possible. This is to ensure the shared codifier may generate significative and easy-to-transfer functions.

However, the problem goes far beyond faces. The neural network is not specifically designed to recognize faces. The various angles and light conditions increase the complexity of the issue, since more steps are required—not only to align faces, but also to identify the relevant features. If the frames used for the training process show a high variability in terms of light conditions, then the codifier is forced to include information on them in the latent representation of the face. This takes a space that may be used to better represent facial expressions, which reduces quality in general. This is called “underfitting” and suggests that the issue is so complex to be completely gathered by the chosen model.

Even if this is true, we must face the opposite problem as well. If all the images provided have the same features, the training process may not gather their real variability. The result is that the neural network will learn to reproduce the same image instead of the same face. This is an issue called “overfitting”.

So, what is better? Shall we include images that are very similar to simplify the problem, or add shots taken in different light conditions to better generalize the appearance of the face? The answer is that it depends. It is impossible to know how the neural network will respond simply by saying that the images are taken under the same light. However, and generally speaking, the more complex a problem is, the harder it is to solve it. Unless you are an expert, you should strive to support the neural network as much as possible.

If your goal is to make a face change in a single clip, you don't need a neural network that is capable of making a face change under any condition. You only need one that works only once. The easiest way to achieve this is to reduce the complexity of the problem. You can start by selecting the correct videos:

- Videos filmed under similar light conditions should be used. Light does not only affect the color of a scene. It also casts shadows on faces that are difficult to match. This is because the reproduction of realistic shadows

requires a three-dimensional understanding of the face geometry, which must be learned on the basis of two-dimensional images.

- Subjects with similar facial features should be chosen. This goes beyond the simple concept of "face". Makeup, glasses and beard make the problem much more difficult. It is particularly difficult to use faces with features outside the face itself (such as long beards or occlusive hair), as they will be trimmed.
- Of course, skin color should be similar. One of the biggest disadvantages of exchanging faces is how faces, once molded by the neural network, are mixed again in the video. The way it works simply softens the edges of the image in a fairly primitive way. If A and B subjects have very different skin colors, it will be difficult to fuse them without problems. Note that some face change implementations adjust the colors of the reconstructed face to make it match the original skin color. Unfortunately, this operation is performed over the whole image and will even affect areas that do not need to be improved (such as eyes and facial hair).

5. Faceswapping with GANs

The capacity of generative models for face generation has been widely confirmed. However, with the VAEs-based approach we cannot make these face changes in real time. This section shows how to use a Generative Adversarial Network so that a model learns facial landmarks detected by a webcam and that these are translated into a face by using the pix2pix library.

The first thing to do is to obtain a set of data to begin training the generative model. Training data should be obtained by getting all possible frames out from each video, as well as key points from the victim's face, commonly referred to as facial landmarks. For this, the dlib estimator was used, which can detect up to 68 reference points (mouth, eyebrows, eyes, etc.) on a face together with OpenCV to process the video file, as shown in Figure 19. The steps to perform and recommendations for appropriate face collection are as follows:

- Detecting facial landmarks is a two-stage process. First, a face detection routine is used to detect the faces and then the dlib estimator is applied to the detected face.
- The estimation of face position is an implementation derived from the investigation.
- It is recommended that the face of the victim to impersonate does not move too much, as well as that the videos intended to collect the victim's faces do not contain a changing or very complex background.

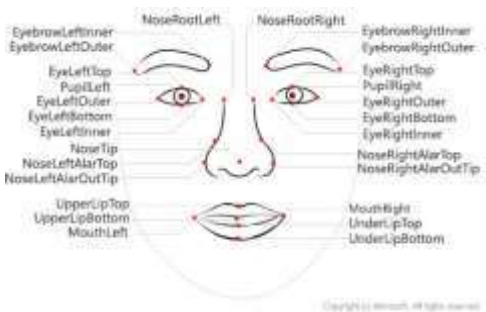

Fig. 19. Main points of interest of Face Landmarks [19]

The face2face-demo repository was used [20], which takes the facial landmarks of all the faces got out from a video for face generation using a Deep Convolutional Generative Adversarial Network (DCGAN).

This repository, which was modified following the instructions left by Karolmajek to increase the resolution of the images to train and test, allows the training of a generative model using the pix2pix-tensorflow repository (original implementation in Torch) [21].

The result of the training of this model, in a system with GPU, was successful after 3 days. Once the model has been trained and before getting on with the testing phase:

- The trained model is reduced (an image is used as an input tensor)
- The model is frozen to a single file

What does it mean to reduce and freeze a model in Machine Learning? When we want to put a model into production, we don't need the metadata attached to our checkpoint, we just want our model and its weights to be well packaged in a file. This facilitates storage, versions and updates of its different models. In this way we can execute the test by providing webcam images as input.

The use of GANs for impersonation is a booming field that, without a doubt, in the following months to the publication of this paper will attract great attention. As we have seen so far, a large number of victim's picture samples are needed for both face rendering and Faceswapping. However, in many practical scenarios, there is not a large dataset, but sometimes just a single image. [21] presents a system that performs an extended meta-learning over a large set of video data, and after that it can frame the learning of few people from models of neural faces of people never seen before as problems of adverse training with high capacity generators and discriminators. The system can initialize the parameters of both the generator and the discriminator in a specific way of the person, so that the training can be based on just a few images and be quickly performed, despite the need to align tens of millions of parameters. With this publication we seek to prove that this approach is capable of modeling new people and even portraits, highly realistic and personalized [22].

6. Audio Generation

With the resources of Faceswapping in investigated videos, we would already be able to generate a non-legitimate multimedia resource that could help anyone impersonate a victim. However, an important resource to complete the attack is missing: the generation of audio with the tone and the inflections of the person to impersonate.

Personalized voice model of the person to impersonate. For this purpose, there are multiple resources that allow a model to generate audio samples from written text (Text-to-Speech), this model having been constructed with audio samples of the person to impersonate along with their transcripts.

Given the digital exposure we face, it is not only easy to obtain videos of people of certain importance that may become the focus of this type of attacks, but to obtain audios from their voices is almost immediate as well thanks to podcasts or YouTube. In this investigation, we used Microsoft Neural Text-to-Speech (TTS): Deep Neural Networks to improve pronunciation and intonation. This text-to-speech resource uses deep Neural Networks, which allows to exceed the limits of traditional text-to-speech systems to match the patterns of accent and intonation in spoken language, called prosody, and to synthesize speech units in a computer voice.

Traditional text-to-speech systems divide prosody into separate linguistic analyses and acoustic prediction steps that are governed by independent models. This can result in a muted and loud voice synthesis. This new service performs simultaneously prosody prediction and voice synthesis. The result is a more fluid and natural sound voice.

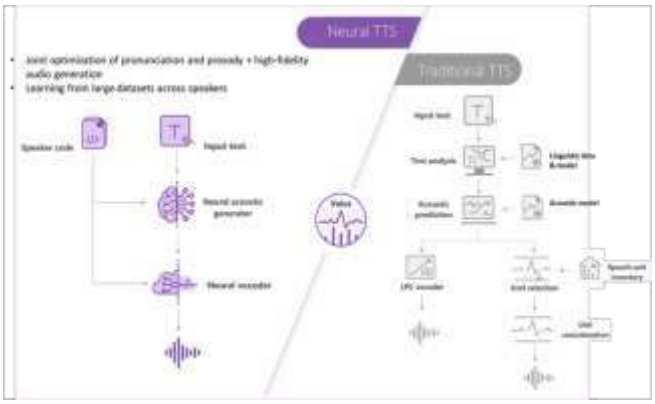

Fig. 20. Azure Text-to-Speech Neural Service for Speech Synthesis [23]

This voice customization allows you to create a recognizable and unique voice for your brand. To create your custom voice source, make a studio recording and load the associated scripts as training data. The service then creates a unique voice model tuned to your recording. You can use this custom voice source to synthesize speech. If you wish to use Neural TTS, you must have an Azure account.

A voice training data set consists of a set of audio files, along with a text file that contains the transcripts of the audio files. It is essential to check the transcripts of these audios to perform a proper training. To produce a good voice source, recordings should be carried out in a quiet room with a high-quality microphone, in order to obtain a good signal-to-noise ratio. Consistent volume, speech speed, speech tone and expressive speech gestures are essential to build a good digital voice. This is often not possible, since you cannot usually get good recordings from the person to impersonate. Normally, digital voice samples are used, which may not necessarily be of good quality or be in the mother tongue of the impersonated. This might entail imperfections in the generated voice due to lower SNR than recommended, or smaller pronunciation scores than desired.

The first step was to isolate audio samples (in English) from the person to impersonate. At the date of this publication, the only languages supported were English and Chinese.

1. Audios (.wav) are downloaded from YouTube in English
2. In total, 314 mono-channel utterances of 30s each were obtained, equivalent to 2.61h of recordings

Fig. 21. Dataset of utterances and transcripts generated and uploaded to the Custom Voice service

3. We needed the transcripts of the audios obtained. For this purpose, Google Cloud Speech-to-Text API was used as it is a free resource with a very high frequency of use. Thanks to this, transcripts in .txt format were obtained.
4. Upload audios and transcripts to train the algorithm and get a strengthened model.
5. Once a successfully constructed model was obtained, we went ahead with its deployment it to test.

The approximate training time of our Custom Voice Font was around 8 hours.

Fig. 22. Result after training our Custom Voice model

The result of the voice was good despite having just a third of the time recommended by Microsoft to have a quality font. The tone was well crafted, but intonation was still robotic. It was concluded that performing a voice postprocessing was essential, being necessary to address voice features such as tone and vibrato, in addition to inflection. We also added a telephone filter and background noise to add more credibility to the generated text.

7. Protection tools

As we have seen so far, the power of deep generative models has significantly improved the quality and efficiency in the generation of realistic-looking fake face videos.

Until a few years ago, it was very easy to distinguish a video on which Faceswapping technology had been applied. However, this is becoming more complicated. Quality evolution in the generation of fake videos is exponential. It is increasingly difficult to distinguish a modified video from a real one. If we also add a voice support to increase the probability that the attack target creates deception, how to protect ourselves from such attacks?

When conducting the investigation, we found the following publication [24]. Along this work, a new method to expose videos of fake faces generated with Neural Networks is described. The method is based on the detection of eye blink in videos, which is a physiological signal not well presented in synthesized fake videos. This method is tested on the landmarks of the blink detection data sets and also shows a promising performance in detecting videos generated with DeepFakes.

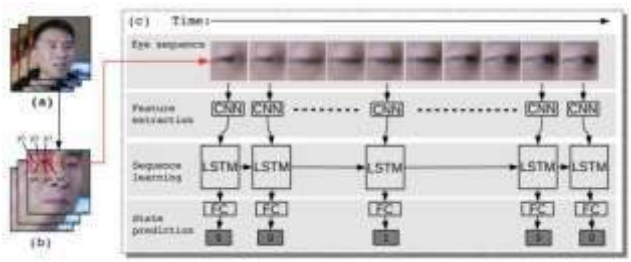

Fig. 23. Convolutional recurrent architecture used to detect blinks in [24]

The repository with the code is available on GitHub [26] and, although it is still under construction, it can be used as a valid example of the application of recurrent convolutional Neural Networks for video analysis. From the Crazy Ideas team we tried to reproduce how the tool works, but we did not have the original dataset that would allow us to calculate the probability that a given video was fake or real, so we decided to look for an alternative, as shown below.

Generally, for a healthy adult person there is an interval of 2-10 seconds between each blink, but the actual rates vary according to the individual and the activity they are performing. The average blink speed at rest is *17 blinks / min or 0,293 blinks / min* (during the conversation, this rate increases to *26 blinks / min*, and decreases to *4,5 blinks / second*; the interpretation of this difference may be interesting in our analysis, since many of the politicians read their speeches when they are supposed to be speaking while they are being filmed). The duration of a blink can range from *0,1 – 0,4 seconds / blink*. [27]

Therefore, we will take as a normal blink landmark a duration in the range *0,1 – 0,4 s* and a blink rate that ranges between *17-26 blinks / min*. To unmask fake videos, we decided to create a very simple Gaussian classifier with Scikit-Learn based on two easily collectable features of a video by using the dlib and opencv libraries [28]:

- Flashing speed [blinks / s]
- Flash duration [s]

We generated random samples for training and testing the model, which would allow us to obtain a map of probabilities that the video was labeled as Real or Fake. In this case, we sought to carry out a simple binary logistic regression (*x* for fake video/ *y* for real video).

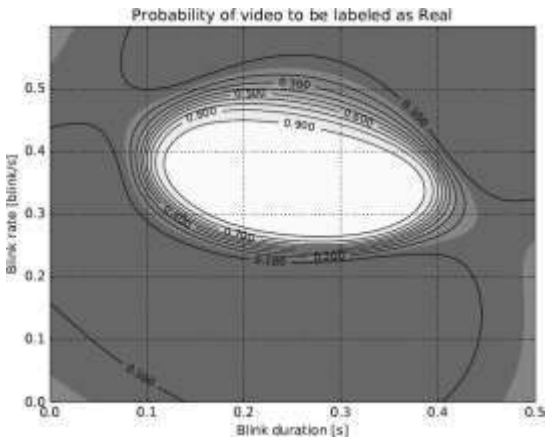

Fig. 24. Probability map of a Gaussian classifier based on blink rate and duration

Once we had our model trained, we only had to test it with the videos we wanted to unmask. Despite the simplicity of the model and the few features collected, the model is capable of discretizing whether a video is fake or not quite rightly. This shows that, although the model could be improved by including additional features, just with minimal notions of Machine Learning we could try to protect ourselves against such attacks.

Social networks have dramatically changed the way news is generated, disseminated and consumed by society, opening unexpected opportunities, but also creating complex challenges. Social networks, digital newspapers, although they are very important sources of information, allow the generation of highly skewed information, as well as to facilitate its rapid dissemination. Making fraudulent use of them, campaigns of non-legitimate information that can affect the credibility of the entire news ecosystem can be easily generated.

An exclusive feature of social media news is that anyone can register as a news editor without any restrictions (for example, anyone can create a Facebook page that claims to be a newspaper or a media organization). Consequently, many traditional digital media are increasingly migrating to social networks. Along with this transition, it is not surprising that there is a growing concern about "fake" news editors who publish "fake" news, and who often disseminate it widely using "fake" followers. As the widespread dissemination of fake news can have a serious negative impact on individuals and society, the lack of scalable fact-finding strategies is especially worrying.

Not surprisingly, recent research efforts are intended not only to better understand this phenomenon, but also to automate the detection of fake news. While a fully automated approach to the problem of fake news can be quite controversial and is still open to debate, we can ask ourselves a relevant question: What is the prediction performance of current approaches and functions for automatic detection of fake news?

Most of the existing efforts in this space are simultaneous works, which identify recurrent patterns in fake news once they have already been disseminated, or propose new features for classification engines.

Therefore, it is difficult to assess the potential of supervised models trained on the basis of the characteristics proposed in recent studies to detect fake news. This paper briefly examines two services that, by using supervised learning classifiers, will allow us to classify the nature of the digital content we consume.

When we start investigating fake news, we can see that there are many different categories into which classify non-true information. There are articles that are blatantly false and provide a true event but then make some false interpretations. They may be classified as pseudoscientific. Actually, these articles are only opinion disguised as news, satirical articles and articles that consist mainly of tweets and quotes from other people. That is, they can be classified as "satire", "false", "deceptive", "clickbait", etc.

Fake news detection is a supervised learning problem on which Natural Language Processing (NLP) techniques are applied. In this case we find web content that can be obtained through web scrapping, and each news can be labeled based on as many criteria as desired.

One of the conditions for fake news classifiers to achieve good performance is to have enough labeled data. However, getting labels to trust requires a lot of time and cataloging work. Therefore, semi-supervised and unsupervised methods can also be used for the grouping of news in a first exploratory approach.

Fig. 25. Classification of skewed text as a task within supervised paradigm

There are several services for the detection of Fake News based on:

- Verification of origin domains as legitimate or poorly skewed source;
- Analysis of patterns in the content of the news;
- Keyword search to improve search, categorization and information management;

The most commonly used classification models in the detection of fake news are Support Vector Machine (SVM) and Naive Bayes Classifier (NBC). Logistic Regression (LR) and decision tree models are also used.

Deep Learning also has a place. Many types of Neural Network models, such as multilayer perceptron, also work for the detection of false news. Recurrent Neural Networks (RNN) are very popular in natural language processing, especially in Long Short-Term Memory (LSTM), as they can capture long-term dependencies.

In the crazy ideas department, we sought to see if it was easy to protect ourselves from this kind of news, so we started to investigate. We found two services chosen for the processing and classification of news: Robinho and fakebox. Later, we generated a small Python script that performed scrapping on a web page. With the URL, the script got all the text from the web page. The two services on which our program worked are briefly described below:

- Robinho

Robinho Fake News Detector allows you to detect and point out Fake News, Clickbaits and extremely skewed news.

There are several ways to use the Fake News Detector [29]:

- Install the extension for Chrome or Firefox. This verifies the news from the Twitter and Facebook feed.
- Talk directly with Robinho on Telegram.
- Call the JSON API, which is what has been done for this paper.

- Fakebox

Fakebox analyzes news articles to assess whether they are likely to be real news or not. By examining a number of available aspects of an article (title, content and URL) using integrated machine learning models, a Fakebox can successfully identify fake news with accuracy greater than 95%. Fakebox checks the following aspects of an article:

- Title: Titles can be clickbait or skewed.
- Content: the textual content of an article can be analyzed in order to determine if it is written as a real news or not.
- Domain name: some domains are known for hosting certain types of content, Fakebox knows the most popular sites thereon.

The generated script, called fake.news.detector.py, can be checked out in the following repository [30].

As you can see, the script is very simple, not only regarding its implementation but also concerning its use. Performing a GET and POST to the following URL dictionary:

```
url_main_dict= {

 'robinho':'https://robinho.fakenewsdetector.org/predict',

 'fakebox': 'http://localhost:8080/fakebox/check'
```

Providing the following parameters in each call:

```
params = dict(url=url, content=parrafos, title=title)
```

You can get the result of both services in JSON format to easily access the data.

8. Conclusions

Along this publication we have tried to go through the state of the art and the most popular use cases of technologies that, by using Machine Learning and Deep Learning (simple models of ML, VAEs, GANs, Convolutional and Recurrent Neural Networks), can not only generate phishing attacks or Fake News, but also protect us from them.

Fake News and disinformation campaigns have become a real problem nowadays. According to Barack Obama:

If everything seems to be the same and no distinctions are made, then we won't know what to protect. We won't know what to fight for. And we can lose so much of what we've gained in terms of the kind of democratic freedoms and market-based economies and prosperity that we've come to take for granted

It is increasingly difficult to discern what information or multimedia resource is real. And as time goes by, generation techniques improve their performance, so their detection will be increasingly difficult over time.

As it has been shown, deceiving people can be 'very cheap'. Anyone with a minimum computer and math knowledge can acquire a fairly powerful computer allowing them to train models such as those described in sections V, VI and VII. You don't even need to buy a device, since you could use Cloud services (e.g., Google Cloud, Amazon Web Service, Microsoft Azure, etc.) with which to create virtual instances that make use of powerful GPUs or even TPUs (such as those provided by Google Cloud). The cost of these services can range from a few hundred to less than € 5,000 per month, an amount that is not negligible at all. However, such amount could be profitable if a small percentage of the phishing campaigns launched were successful.

Protecting yourself is also "cheap". With basic knowledge of Machine Learning we can protect ourselves against both skewed or illegal text and against videos modified for spurious purposes, as shown in section 8.

It seems feasible that Artificial Intelligence and Cybersecurity will go hand in hand in the future. Although AI limits have not yet been discerned, the attack capabilities of relatively simple models have been confirmed, making use of resources that are easily accessible from multiple repositories. For such reason, building defenses that protect us against these attacks will become a necessity.

We live in an era of over information. We can access any information resource from wherever we are in the world with a simple click. The current ease for producing and disseminating content constitutes great news for general knowledge, but it also fosters the generation of disinformation or deception campaigns, whose volume and sophistication becomes so high that it is impossible to delimit its scope on society. Therefore, the best tools to protect ourselves will continue to be knowledge, embracing education and awareness, as well as critical sense—always questioning the truthfulness of both the sources and the information consumed.

9. References

[1]"Fake Obama created using AI tool to make phoney speeches" Fake Obama created using AI tool to make phoney speeches

https://www.bbc.com/news/av/technology- 40598465/fake-obama-created-using-ai-tool-to-make-phoney-speeches

[2]"El vídeo que pone el rostro de Steve Buscemi en el cuerpo de Jennifer Lawrence es un recordatorio del peligro de los deepfakes" El vídeo que pone el rostro de Steve Buscemi en el cuerpo de Jennifer Lawrence es un recordatorio del peligro de los deepfakes

https://www.xataka.com/robotica-e-ia/video-que-pone-rostro-steve-buscemi-cuerpo-jennifer- lawrence-recordatorio-peligro-deepfakes

[3]"Machine Learning y Deep Learning: cómo entender las claves del presente y futuro de la inteligencia artificial" Machine Learning y Deep Learning: cómo entender las claves del presente y futuro de la inteligencia artificial

https://www.xataka.com/robotica-e-ia/machine-learning-y- deep-learning-como-entender-las-claves-del-presente-y- futuro-de-la-inteligencia-artificial

[4]"Machine learning can change the way institutions operate" Machine learning can change the way institutions operate

https://www.ellucian.com/insights/machine- learning-can-change-way-institutions-operate

[5]"Aprendizaje Supervisado (Wikipedia)" Aprendizaje Supervisado (Wikipedia)

https://es.wikipedia.org/wiki/Aprendizaje_supervisado

[6]"Algoritmo (Wikipedia)" Algoritmo (Wikipedia)

https://es.wikipedia.org/wiki/Algoritmo

[7]"Lesson 07 - Scikit-Learn" Lesson 07 - Scikit-Learn

http://jeremy.kiwi.nz/pythoncourse/2016/04/01/lesson07rd.html

[8]"A (Long) Peek into Reinforcement Learning" A (Long) Peek into Reinforcement Learning

https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek- into-reinforcement-learning.html

[9]"TPU DE CLOUD - Google Cloud" TPU DE CLOUD - Google Cloud

https://cloud.google.com/tpu/

[10]"Single-Layer Neural Networks and Gradient Descent" Single-Layer Neural Networks and Gradient Descent

https://sebastianraschka.com/Articles/2015_singlelayer_neu rons.html

[11]Deep Learning by Adam Gibson, Josh Patterson Publisher: O'Reilly Media, Inc. Release Date: August 2017 ISBN: 9781491924570

[12]"LaTeX code for making neural networks diagrams" LaTeX code for making neural networks diagrams

https://github.com/HarisIqbal88/PlotNeuralNet

[13]"Lecture 13: Generative Models" Lecture 13: Generative Models

http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture 13.pdf

[14]"From Autoencoder to Beta-VAE" From Autoencoder to Beta-VAE

https://lilianweng.github.io/lil- log/2018/08/12/from-Autoencoder-to-beta-vae.html

[15]"Kullback-Leibler divergence (Wikipedia)" Kullback- Leibler divergence (Wikipedia)

https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[16]"Slide 12 in Kingma's NIPS 2015 workshop talk" Slide 12 in Kingma's NIPS 2015 workshop talk

http://dpkingma.com/wordpress/wp- content/uploads/2015/12/talk_nips_workshop_2015.pdf

[17]"From Autoencoder to Beta-VAE" From Autoencoder to Beta-VAE

https://lilianweng.github.io/lil-log/2018/08/12/from- autoencoder-to-beta-vae.html

[18]Generative Adversarial Nets. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. 2014. Departement d'informatique et de recherche operationnelle Universite de Montréal

https://arxiv.org/pdf/1406.2661.pdf

[19]"Face detection and attributes" Face detection and atributes

https://docs.microsoft.com/bs-latn- ba/azure/cognitive-services/face/concepts/face-detection

[20]"pix2pix demo that learns from facial landmarks and translates this into a face" pix2pix demo that learns from facial landmarks and translates this into a face

https://github.com/datitran/face2face-demo

[21]Few-Shot Adversarial Learning of Realistic Neural Talking Head Models. Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, Victor Lempitsky. 20 May 2019.

https://arxiv.org/abs/1905.08233

[22]"Few-Shot Adversarial Learning of Realistic Neural Talking Head Models." Few-Shot Adversarial Learning of Realistic Neural Talking Head Models.

https://t.co/Xk5D4WccpD

[23]Image-to-Image Translation with Conditional Adversarial Networks. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei

A. Efros; Nov 2016.

[24]"Microsoft's new neural text-to-speech service helps machines speak like people" Microsoft's new neural text-to-speech service helps machines speak like people

https://azure.microsoft.com/es-es/blog/microsoft-s-new- neural-text-to-speech-service-helps-machines-speak-like-people

[25]In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. Yuezun Li, Ming-Ching Chang and Siwei Lyu. 2018. Computer Science Department, University at Albany, SUNY.

https://arxiv.org/pdf/1806.02877.pdf

[26]"In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking Repository" In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking Repository

https://github.com/danmohaha/WIFS2018_In_Ictu_Oculi

[27]Analysis of blink rate patterns in normal subjects. Bentivoglio AR, Bressman SB, Cassetta E, Carretta D, Tonali P, Albanese A. National Center for Biotechnology Information 1997 Nov;12(6):1028-34.

[28]"Eye blink detection with OpenCV, Python, and dlib" Eye blink detection with OpenCV, Python, and dlib

https://www.pyimagesearch.com/2017/04/24/eye-blink- detection-opencv-python-dlib/

[29]"Flag and Detect Fake News with the help of AI" Flag and Detect Fake News with the help of AI https://fakenewsdetector.org/ https://github.com/fake-news- detector/fake-news-detector

[30]"Custom Fake News Detector Repository for RootedCON 2019" Custom Fake News Detector Repository for RootedCON 2019

https://github.com/eblancoh/fakenews

About ElevenPaths

At ElevenPaths, the Telefónica's Cybersecurity Unit, we believe in the idea of challenging the current state of security, since security constitutes a feature that must be always present in technology. We are continuously redefining the relationship between security and people, with the aim of developing innovative products capable of renovating the concept of security. Thanks to this, we stay a step ahead of attackers, that are increasingly present in our digital life.

More information

www.elevenpaths.com
@ElevenPaths
blog.elevenpaths.com