Eleven Paths

WHITEPAPER_

La Inteligencia Artificial: Aplicabilidad de GANs y Autoencoders en la Ciberseguridad

30.05.2019

Telefónica CYBER SECURITY UNIT

elevenpaths.com

Índex

1. Introducción 3

2. Antecedentes 3

3. Conceptos 4
- 3.1. ¿Qué es el *Machine Learning*? 4
- 3.2. ¿Qué es un modelo de *Machine Learning*? 5
- 3.3. ¿Qué es un algoritmo de *Machine Learning*? 5
- 3.4. Paradigma de aprendizaje supervisado 5
- 3.5. Paradigma de aprendizaje no supervisado 6
- 3.6. Aprendizaje reforzado 7
- 3.7. ¿Qué es entrenar en *Machine Learning*? 7
- 3.8. ¿Qué es *Deep Learning*? 8
- 3.9. La neurona y el perceptrón 9
- 3.10. ¿Qué es una red neuronal? 10
- 3.11. Teoría de Modelos Generativos 11
- 3.12. Variational Autoencoders 11
- *3.13.* Reparametrización *trick* 15
- 3.14. Entrenamiento de una GAN 17

4. Faceswapping con VAEs 18

5. *Faceswapping* con GANs 20

6. Generación de audio 22

7. Herramientas de protección 24

8. Conclusiones 27

9. Referencias 29

Acerca de ElevenPaths 32

Más información 32

Resumen ejecutivo

La inteligencia artificial y la ciberseguridad son dos pilares fundamentales en el proceso de transformación digital de la sociedad y de las organizaciones. El día a día de esta revolución digital, que ha cambiado la forma en la que el ser humano se comunica, se relaciona, trabaja y vive, pasa por conocer los riesgos que las nuevas tecnologías traen consigo. Cualquier usuario puede entender los beneficios que puede proporcionar la inteligencia artificial, pero debemos ser cautos y conocer los riesgos que ésta puede proporcionar. En este artículo se habla del *Machine Learning* y *Deep Learning* mostrando diferentes caminos. La posibilidad de que un atacante pueda suplantar identidades, voces, imágenes, videos a través de una IA supone un gran riesgo para la sociedad como para las organizaciones. En este artículo se muestran la aplicabilidad de la inteligencia artificial para protección y su cara más ofensiva.

1. Introducción

El mundo que conoces está en constante cambio. La revolución digital, la transformación digital de las empresas y de la propia sociedad está suponiendo un cambio en la forma de vivir y de relacionarse. Las amenazas del mundo físico tienden a digitalizarse y son éstas las que suponen un aumento en el fraude.

La aplicación de la inteligencia artificial y la evolución de ésta ya se encuentra arraigada en el día a día de la sociedad y en el futuro de las empresas. La utilización de la IA para tareas comunes del día a día es una realidad, por ejemplo, el acceso a una plataforma de compras que aprende que tipo de gustos tienen sus clientes y le ofrecen los productos relacionados con ello.

La Inteligencia Artificial, el *Deep Learning*, *el Machine Learning* son términos que están aquí y que se encuentran en constante evolución. Facilitar la vida de las personas puede ser uno de sus objetivos, aunque se pueda hacer un mal uso de ellos. Uno de los ejemplos más claros es la proliferación de *Fake News* apoyadas en videos generados por una IA que parecen ser lo que no son. La modificación de imágenes o videos por una IA para introducir o eliminar ciertos aspectos. La generación de voces por una IA que encajan perfectamente con las de otras personas.

Estos ejemplos son, sin lugar a la duda, casos relacionados con la ciberseguridad y la concienciación que se debe instaurar en la nueva sociedad, la sociedad de la transformación digital.

2. Antecedentes

Respecto a los antecedentes se pueden encontrar diversos ejemplos del uso de la inteligencia artificial para realizar ataques de identidad o realizar fraude contra otros usuarios. Uno de los ejemplos más claros es un video de Barack Obama en el que se puede ver, aparentemente, al ex-presidente de los Estados Unidos dando un mensaje que otra persona quiere.

El video fue publicado el 19 de julio de 2017 [1]. El video se generó utilizando un modelo entrenado para generar la boca y el movimiento de ésta con las palabras que se querían decir. En concreto, se 'mapeaba' el texto a la boca, no real, de Obama. Todo esto era generado, después de un extenso entrenamiento, por una IA. Aquí está el riesgo y la novedad. No es un montaje hecho por una persona, el video está generado por una IA que hace un s*wapping* de la boca de Obama

y genera la boca con el movimiento que se haría al pronunciar el texto. Además, la voz puede ser generada por una IA. Es decir, se puede entrenar a una IA para generar una voz idéntica a la de Obama.

Esto abre un debate y es, ¿Lo que se ve y se escucha es real? ¿Puede la sociedad fiarse de lo que ve y escucha? Esto es un paso más en el mundo de las *Fake News*, ya que abre un mundo de posibilidades. Se puede vislumbrar como una evolución de las técnicas de *phishing* y de ingeniería social, ya que permitirán hacer pensar a cualquier persona que, realmente, es la persona en cuestión quién lanza el mensaje y a la que están viendo.

Otro de los ejemplos de las posibilidades del Faceswapping, independientemente de la técnica o tecnología utilizada, es el de Jennifer Lawrence y Steve Buscemi [2]. En este ejemplo no se pretende mostrar el fraude, si no las posibilidades de la IA al poder intercambiar la cara de una persona en el cuerpo de otra. El video se ha podido generar con Autoencoders y muestra las facciones de la cara del sujeto de origen y el rostro del sujeto que se quiere introducir. Tras un entrenamiento exhaustivo, la IA aprende a generar *frames* con el cuerpo de Jennifer Lawrence, las facciones y gestos de ésta, pero con el rostro o la cara de Steve Buscemi.

3. Conceptos

En esta sección se introducen algunos conceptos básicos de aprendizaje automático que resulta necesario tener claros para poder entender el manejo de las tecnologías involucradas tanto en la generación como en la generación de recursos para protegernos contra este tipo de contenidos.

3.1. ¿Qué es el *Machine Learning*?

Definimos el *Machine Learning* o Aprendizaje Automático como "La capacidad de una máquina de aprender de la experiencia". El objetivo del *Machine Learning* es crear algoritmos que puedan aprender cómo realizar una determinada tarea basándose en datos de ejemplo.

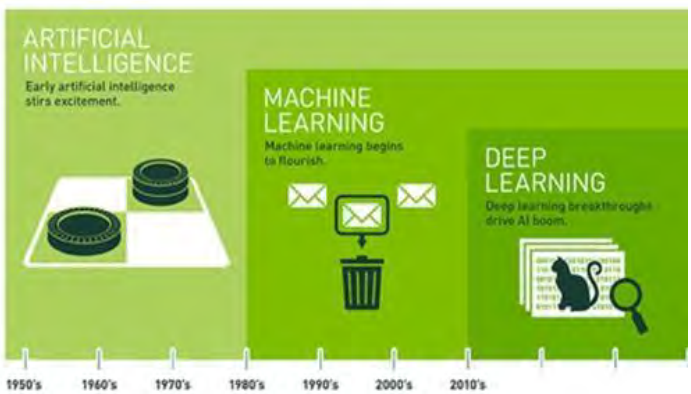

Fig. 1. Relación entre Inteligencia Artificial, Machine Learning y Deep Learning [3]

El *Machine Learning* es una rama de la Inteligencia Artificial que tiene como objetivo extraer la mayor cantidad de información y conocimiento posible de los datos. En los últimos años, el *Machine Learning* se ha convertido en algo prácticamente ubicuo en nuestras vidas, pues está presente en múltiples servicios de los que hacemos uso en nuestra vida cotidiana.

El *Machine Learning* en su uso más básico es la práctica de usar algoritmos para *parsear* datos, aprender de ellos y luego ser capaces de hacer una predicción o sugerencia sobre algo. La máquina es entrenada utilizando una gran cantidad de datos dando la oportunidad a los algoritmos a ser perfeccionados.

3.2.¿Qué es un modelo de *Machine Learning*?

Un modelo de aprendizaje automático es una representación matemática de un proceso del mundo real. Para generar un modelo de *Machine Learning*, deberemos proporcionar datos de entrenamiento a un algoritmo de aprendizaje automático para que el algoritmo pueda "aprender". Un modelo no es más que una función con la capacidad de dar una predicción.

3.3.¿Qué es un algoritmo de *Machine Learning*?

Los algoritmos ML son aquellos que pueden aprender de los datos y mejorar a partir de la experiencia, sin intervención humana. Las tareas de aprendizaje pueden incluir aprender la función que asigna la entrada a la salida o aprender la estructura oculta en datos sin etiquetar. Formalmente, un algoritmo es un conjunto de pasos realizados en orden.

Un modelo de *Machine Learning*, como la regresión o el agrupamiento o las Redes Neuronales, se basan en el funcionamiento de los algoritmos para su ejecución. Los algoritmos son el motor que subyace en el código que ejecutamos. Hacen todo el trabajo pesado de multiplicar matrices, optimizar resultados y generar una salida. Existen muchos tipos de modelos, que abarcan un ecosistema completo de aprendizaje automático de manera más general.

La palabra "modelo" es bastante nebulosa y difícil de separar de algo como una "función" o una "ecuación". Dado que separar los conceptos de modelo y algoritmo parece difícil, vamos a definir un algoritmo de ML como un conjunto de pasos que se pasan a un modelo para que éste realice los cálculos o el procesamiento.

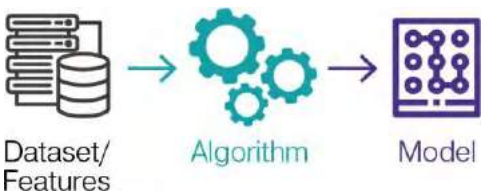

Fig. 2. Pipeline of training a Machine Learning model [4]

3.4.Paradigma de aprendizaje supervisado

Los algoritmos más usados en *Machine Learning* son aquellos capaces de automatizar un proceso tras haber aprendido a partir de un conjunto de ejemplos conocidos. Todos los algoritmos abarcados bajo esta premisa están agrupados como algoritmos de aprendizaje supervisado [5], en el que el usuario facilita a un determinado algoritmo [6] pares entrada- salida de ejemplo, siendo el algoritmo el encargado de aprender la relación para posteriormente ser capaz de dar una salida para una entrada jamás vista sin la supervisión de ningún ser humano. En los problemas de aprendizaje supervisado se enseña o entrena al algoritmo a partir de datos que ya vienen etiquetados con la respuesta correcta. Cuanto mayor es el conjunto de datos más el algoritmo puede aprender sobre el tema. Una vez concluido el entrenamiento, se le brindan nuevos datos, ya sin las etiquetas de las respuestas correctas, y el algoritmo de aprendizaje utiliza la experiencia pasada que adquirió durante la etapa de entrenamiento para predecir un resultado.

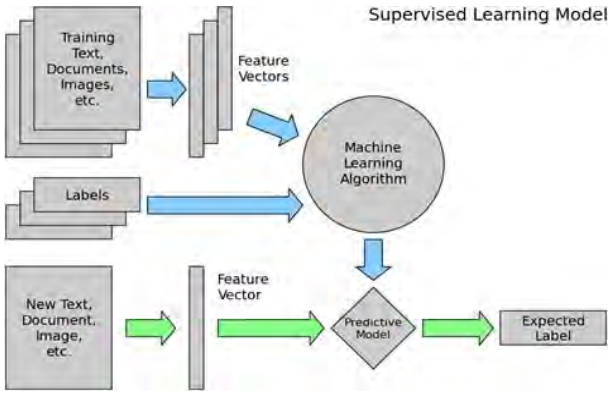

Fig. 3. Diagrama de entrenamiento y testeo bajo paradigma supervisado [7]

Los métodos de aprendizaje supervisado son principalmente de dos clases, según el tipo de tareas de ML que pretendan resolver: clasificación y regresión. Poniendo como ejemplo un detector de spam, al modelo en el que se base ese filtro se le pasará un conjunto de mails maliciosos etiquetados como no legítimos junto con otro conjunto de mails no maliciosos identificados como legítimos. Una vez que ese modelo esté entrenado, será capaz de discernir si un mail recibido, no involucrado en el entrenamiento del modelo, es o no legítimo.

3.5.Paradigma de aprendizaje no supervisado

El otro paradigma corresponde al aprendizaje no supervisado, donde sólo se proporcionan los datos de entrada al modelo. Hay múltiples aplicaciones de este tipo de método, aunque pueden resultar más difíciles o complicadas de evaluar. En los problemas de aprendizaje no supervisado el algoritmo es entrenado usando un conjunto de datos que no tiene ninguna etiqueta; en este caso, nunca se le dice al algoritmo lo que representan los datos. La idea es que el algoritmo pueda encontrar por si solo patrones que ayuden a entender el conjunto de datos. Como símil, el aprendizaje no supervisado es similar al método que utilizamos para aprender a hablar los bebés, quienes en un principio escuchan hablar a la gente que les rodea sin entender en un principio nada. Con el paso del tiempo, escuchando miles de conversaciones, su cerebro comenzará a construir un modelo sobre cómo funciona el lenguaje que permitirá reconocer ciertos patrones y a actuar en función de ciertos sonidos.

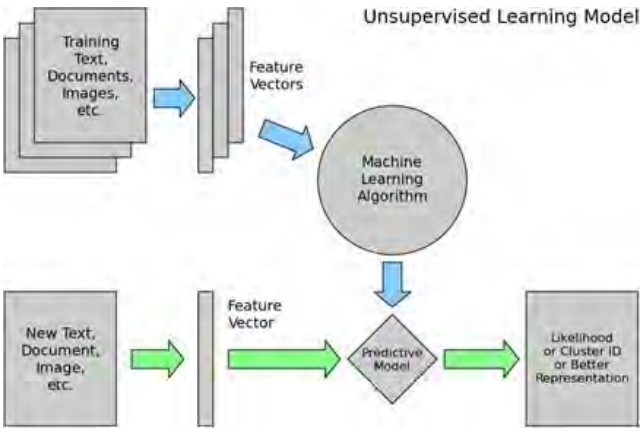

Fig. 4. Diagrama de entrenamiento y testeo bajo paradigma no supervisado [7]

Algunos de los algoritmos más comunes en aprendizaje no supervisado son:

- *Density estimation*: basado en encontrar la función de probabilidad de una variable independiente dado un conjunto de observaciones caracterizadas por esas variables.
- *Clustering*: principalmente dedicado a encontrar los agrupamientos del conjunto de inputs no categorizados que tengan características comunes.
- *Feature Learning*: formado por un conjunto de técnicas que permite que un sistema encuentre las representaciones necesarias para la detección o clasificación de características a partir de datos sin procesar.
- *Dimension reduction*: dedicados a reducir el número de variables aleatorias de una muestra para caracterizar la misma bajo un conjunto de variables principales. Entre los más usados está *Principal Component Analysis* (PCA) y *t-distributed Stochastic Neighbor Embedding* (t-SNE).

3.6. Aprendizaje reforzado

Dentro de estos métodos no supervisados también podemos integrar el Aprendizaje Reforzado. En los problemas de Aprendizaje Reforzado, el algoritmo aprende observando el mundo que le rodea. Su información de entrada es el *feedback* o recompensa de un determinado como respuesta a sus acciones en un determinado entorno. En este caso, el sistema aprende a base de ensayo-error a través de una investigación exhaustiva de las consecuencias de sus acciones en el entorno.

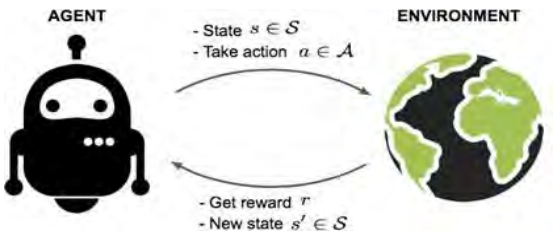

Fig. 5. Diagrama básico de aprendizaje de un agente mediante aprendizaje reforzado [8]

Los principales pasos de un método de aprendizaje de refuerzo se mencionan a continuación:

1. Preparar un agente con un conjunto de políticas iniciales y estrategia.
2. Observar el entorno y el estado actual
3. Seleccionar las políticas óptimas y realizar una acción
4. Obtener una recompensa correspondiente (o una penalización)
5. Actualizar las políticas si es necesario
6. Repetir los pasos 2 a 5 de forma iterativa hasta que el agente aprenda las políticas óptimas.

3.7. ¿Qué es entrenar en *Machine Learning*?

El proceso de entrenamiento de un modelo de ML implica proporcionar a un algoritmo de ML (es decir, el algoritmo de aprendizaje) con datos de entrenamiento para que este pueda aprender, adaptándonos a las condiciones iniciales de disposición de datos que se nos imponen en cada paradigma.

3.8.¿Qué es *Deep Learning*?

Siguiendo la evolución del *Machine Learning* en la última década se ha propagado con más fuerza una técnica concreta de *Machine Learning* conocida como *Deep Learning*.

En los últimos años se han desarrollado el *Deep Learning*, potenciado a través del uso de dispositivos como las unidades de procesamiento gráfico GPUs (*Graphics Processing Units*) y las más recientes TPUs (*Tensor Processing Units*), que han permitido una investigación en el refinamiento de arquitecturas más rápido.

Fig. 6. Cloud TPU v2 Pod Alfa 11.5 petaflops 4 TB de HBM. Red en malla toroidal 2-D. [9]

El cerebro es el órgano más fascinante del cuerpo humano. Gracias a este órgano somos capaces de procesar absolutamente todo lo que nos rodea a través de la vista, el oído, el olfato, el gusto y el tacto. No sólo eso, sino que además nos permite almacenar recuerdos, experimentar emociones, e incluso soñar. Sin el cerebro, seríamos organismos primitivos, incapaces de realizar cualquier acción más compleja que el más simple de los reflejos. El cerebro es, indefectiblemente, lo que nos hace inteligentes.

El cerebro infantil solo pesa medio kilo, pero de alguna manera es capaz de resolver problemas complejos que resultan prácticamente imposibles para los más potentes superordenadores. En cuestión de meses, esos pequeños cerebros son capaces de reconocer caras, discernir objetos a grandes distancias, distinguir e identificar voces, comenzando a asociar sonidos con significados específicos. En la infancia temprana, el cerebro va incorporando de manera exponencial nuevas capacidades, adquiriendo una comprensión sofisticada de la gramática e incorporando miles de palabras en sus vocabularios.

Durante décadas, hemos soñado con construir máquinas inteligentes con cerebros similares al nuestro: asistentes robóticos para limpiar nuestros hogares, coches con conducción autónoma, sistemas que detectan automáticamente enfermedades con una tasa de acierto por encima de la humana... y es algo que estamos consiguiendo. Pero desarrollar y construir estas máquinas supone enfrentamos a nuevas problemáticas que nos obliga a resolver problemas computacionales complejos para tareas que a nuestro cerebro sólo le lleva resolver en microsegundos. Para abordar ciertos tipos de problemas, nos hemos visto obligados esperar la llegada de nuevas tecnologías, a desarrollar una forma de programación radicalmente diferente y a desarrollar nuevos algoritmos que puedan acelerar la resolución de ciertas tareas.

¿Por qué ciertos problemas resultan tan difíciles de resolver para los ordenadores? Los ordenadores tradicionales están diseñados para ser muy buenos en dos aspectos: 1) realizar cálculos aritméticos muy rápidos y 2) realizar esos cálculos siguiendo una lista de instrucciones explícitamente indicadas.

Por definición, *Deep Learning* es un subconjunto dentro del campo del *Machine Learning*, En *Deep Learning*, en lugar de enseñarle a ordenador una lista enorme de reglas para solventar un problema, le damos un modelo que pueda evaluar ejemplos y una pequeña colección de instrucciones para modificar el modelo cuando se produzcan errores. Con el tiempo esperamos que esos modelos sean capaces de solucionar el problema de forma extremadamente precisa, gracias a que el sistema es capaz de extraer patrones. Aunque existen distintas técnicas para implementar *Deep*

Learning, una de las más comunes es simular un sistema de redes artificiales de neuronas dentro del software de análisis de datos.

Las Redes Neuronales son un modelo computacional que comparte algunas propiedades con el cerebro animal en el que muchas unidades simples trabajan en paralelo sin una unidad de control centralizada. Los pesos entre las unidades son el principal medio de almacenamiento de información a largo plazo en Redes Neuronales. Actualizar las ponderaciones es la principal forma en la que una red neuronal es capaz de aprender. El comportamiento de una red neuronal queda definido por la arquitectura de su red. La arquitectura de una red neuronal queda en parte caracterizada por lo siguiente:

- El número de neuronas
- Su profundidad o número de capas
- Las conexiones existentes entre capas

3.9. La neurona y el perceptrón

La unidad básica del cerebro humano es la neurona. Una pequeña parte del cerebro, aproximadamente del tamaño de un grano de arroz, contiene más de 10,000 neuronas, cada una de las cuales tiene un promedio de 6,000 conexiones con otras neuronas. Es esta red biológica masiva la que nos permite experimentar el mundo que nos rodea. Nuestro objetivo en esta sección será imitar esta estructura para usarla en la construcción de modelos de modelos de aprendizaje automático que resuelvan problemas en una forma análoga.

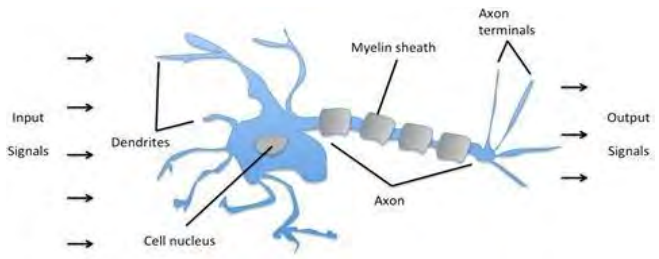

Fig. 7. Schematic of a Biological Neuron. [10]

En su núcleo, la neurona está optimizada para recibir información de otras neuronas, procesar esta información de una manera única, y enviar su resultado a otras células nerviosas. Este proceso queda sintetizado en la Figura. La neurona recibe sus entradas a lo largo de estructuras similares a antenas. llamadas dendritas. Cada una de estas conexiones entrantes se fortalece dinámicamente o debilitado en función de la frecuencia con que se usa, y es la fuerza de cada conexión lo que determina la contribución de la entrada a la salida de la neurona. Después de ser ponderados por la fuerza de sus respectivas conexiones, las entradas se suman juntas en el cuerpo de la célula. Esta suma se transforma entonces en una nueva señal que se propaga a lo largo del axón de la célula y se envía a otras neuronas.

Las neuronas biológicas individuales parecen comportarse de una manera bastante simple, pero están organizadas en una vasta red de miles de millones de neuronas, cada neurona típicamente conectada a miles de otras neuronas. Cálculos altamente complejos pueden ser realizados por una vasta red de neuronas bastante simples. En conjunto, todo el cerebro funciona de esta manera, y lo que es más importante: de una manera descentralizada, en las que todas las partes están comunicadas entre sí mientras realizan cálculos en paralelo.

Warren McCulloch y Walter Pitts (1943) propusieron un modelo muy simple de la neurona biológica, que luego se conoció como una neurona artificial: tiene una o más entradas binarias (on/off) y una salida binaria. McCulloch y Pitts demostraron que incluso con un modelo tan simple, era posible construir una red de neuronas artificiales con la capacidad de calcular cualquier proposición lógica. A partir de aquí surge el concepto de perceptrón. El perceptrón es un clasificador binario con una relación de entrada-salida simple como en el que se realiza un producto escalar de un n

número de entradas con sus pesos asociados y luego enviamos esta "entrada de red" a una función de paso con un umbral definido. Esta función de paso o función de activación, suele ser una función de escalón de Heaviside con un valor de umbral de. Esta función generará un único binario de valor real, dependiendo de la entrada.

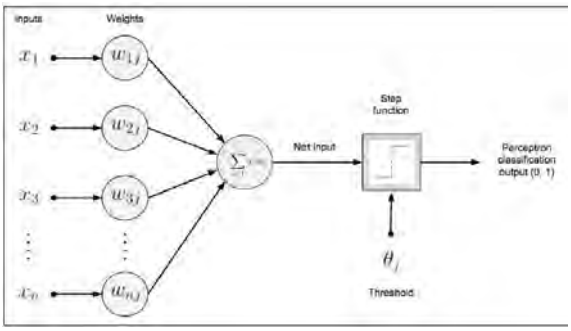

Fig. 8. Single-layer perceptron [11]

3.10.¿Qué es una red neuronal?

La red neuronal más conocida y más fácil de entender es la red neuronal multicapa (*feedforward multilayer neural network),* haciendo un *stacking* de perceptrones. Cuenta con una capa de entrada, una o muchas capas ocultas y una sola capa de salida. Cada capa puede tener un número diferente de neuronas y cada capa está completamente conectada a la capa adyacente. Las conexiones entre las neuronas en las capas forman un gráfico acíclico, como se ilustra a continuación.

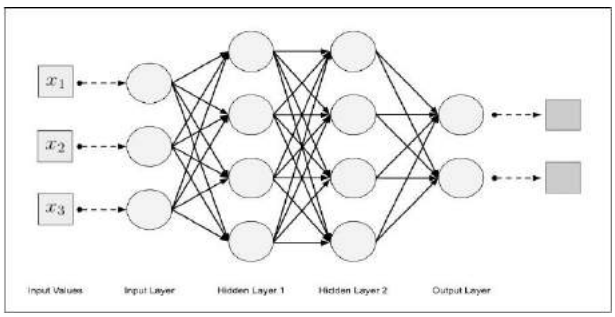

Fig. 9. Multi-layer neural network topology [11]

Una red neuronal multicapa puede representar cualquier función siempre que cuente con suficientes unidades de neuronas artificiales. En general, se entrena mediante un algoritmo de aprendizaje.

Los algoritmos de aprendizaje (asociados a modelos concretos de redes) permiten ir modificando los pesos de las conexiones sinápticas de forma que la red aprenda a partir de los ejemplos que se le presentan.

En el caso del *Deep Learning*, se trata de proporcionar la suficiente cantidad de datos a las capas de neuronas para que puedan reconocer patrones, clasificarlos y categorizar. Una de las grandes ventajas es trabajar a partir de datos no etiquetado y analizar sus patrones de comportamiento y ocurrencia.

Por ejemplo, puedes tomar una imagen como información de entrada de la primera capa. Allí será particionada en miles de trozos que cada neurona analizará por separado. Podemos analizar el color, la forma, etc. Cada capa es experta en una característica y le va asignando un peso. Finalmente, las capas del final de neuronas recogen esa información y ofrece un resultado.

3.11.Teoría de Modelos Generativos

Un Modelo Generativo tiene como misión aprender cualquier tipo de distribución de datos a través de un aprendizaje no supervisado. Todos los tipos de modelos generativos apuntan a aprender la verdadera distribución de datos del conjunto de entrenamiento y, una vez logrado, ser capaz generar nuevas muestras con ligeras variaciones. No siempre resulta posible abstraer la distribución exacta de nuestros datos, por lo que, haciendo uso de las Redes Neuronales, se intenta modelar una distribución lo más similar posible a la distribución verdadera.

Mientras que un modelo discriminativo es un modelo de probabilidad condicionada de un target dada una variable independiente x, que matemáticamente queda representado como. Los modelos generativos permiten calcular la distribución de probabilidad conjunta entre un observable y una variable tal que.

En los últimos dos arquitecturas profundas se han erigido como los principales pilares para desarrollar modelos generativos: *Variational Autoencoders* (VAEs) y *Generative Adversarial Networks* (GANs). Los VAEs intenta maximizar la similitud entre las imágenes generadas y las imágenes con las que han sido alimentadas registro de datos y las GANs se entrenan con la intención de lograr un equilibrio entre un generador y un discriminador.

3.12.Variational Autoencoders

En contraste con los usos más estándar de las Redes Neuronales como modelos para realizar regresión o clasificación, los VAEs son poderosos modelos generativos, que en los últimos tiempos están encontrando aplicaciones en campos muy diversas que van desde generar rostros humanos hasta a producir muestras de audio sintético.

Las Redes Neuronales pueden adquirir infinitas configuraciones en materia de forma y tamaños. Y es exactamente esa forma y el tamaño lo que determina el rendimiento de la arquitectura para resolver un determinado problema.

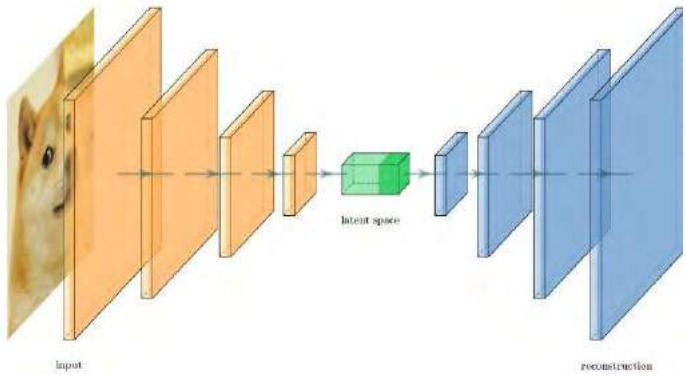

Fig. 10. Arquitectura típica de un Autoencoder [12]

Un *Autoencoder* es un tipo especial de red neuronal cuyo objetivo es hacer coincidir la entrada que se le proporcionó con la mayor semejanza posible. A primera vista, no parecen resolver ningún problema real, pero el valor de este tipo de arquitecturas reside en las características que se pueden extraer de la abstracción de las entradas que realizan en las capas profundas de su arquitectura. El *Autoencoder* debe comprimir la información para reconstruirla a posteriori.

Tras un entrenamiento correcto, el *Autoencoder* aprenderá a representar los valores de entrada en una forma diferente, pero mucho más compacta. Generalmente en una red neuronal convencional, uno intenta predecir un vector objetivo a partir de los vectores de entrada. En un *Autoencoder*, uno intenta predecir a partir de. Es trivial aprender una asignación asi la red no tiene restricciones, pero si la red está restringida, el proceso de aprendizaje se vuelve más interesante.

El *Autoencoder* (AE) más simple tiene una estructura tipo MLP (*Multi Layer Perceptron*). Los *Autoencoders* no requieren de ninguna etiqueta, pueden ser entrenados bajo el paradigma no supervisado.

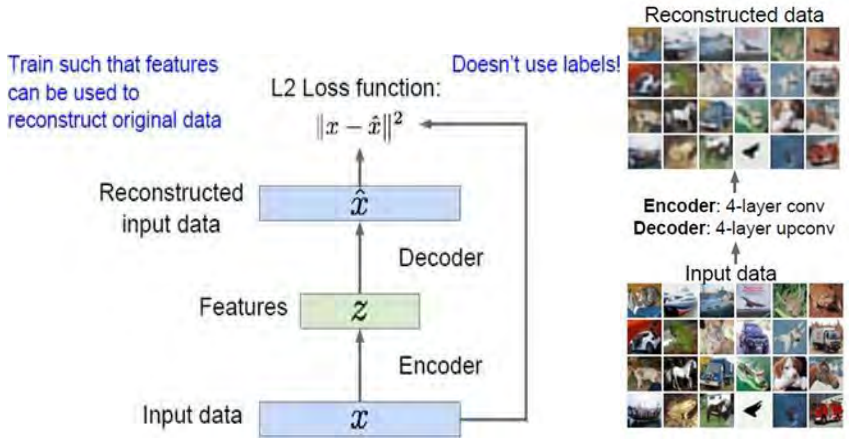

Fig. 11. Arquitectura y funcionamiento de un Autoencoder [13]

La única restricción con la que cuentan este tipo de arquitecturas radica en que la dimensionalidad de la capa oculta es siempre menos que el tamaño de la de entrada. De esta forma se limita la cantidad de información que fluye a través de la red neuronal, obligando al *Autoencoder* a aprender sólo aquellos atributos realmente importantes de los datos de entrada que le permitan reconstruir el estado de entrada a partir de un estado codificado. Idealmente, durante la compresión de la imagen el modelo aprenderá a extraer los atributos latentes de los datos de entrada.

El diagrama anterior muestra una imagen que se alimenta a un *encoder*. Su resultado es una representación dimensional inferior de esa misma imagen, que a veces se denomina vector base o imagen latente. Dependiendo de la arquitectura de la red, la cara latente puede no parecer una cara en absoluto. Cuando se pasa a través de un *decoder*, la cara latente se reconstruye. Los *Autoencoders* tienen pérdidas, por lo que es poco probable que la cara reconstruida tenga el mismo nivel de detalle que estaba presente originalmente.

Matemáticamente, un *Autoencoder* puede definirse como:

- Un input *x*.
- Una función de activación aplicada a cada neurona que se encuentre en las capas ocultas *a(h)*.
- $W_i \in R^{I \times O}$ como la matriz de pesos de la capa i- ésima con dimensión, siendo I el tamaño del input y O el tamaño del output
- $b_i \in R^O$ como la matriz de bias de cada cap a

$$z = a\ xW_1 + b_1$$

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

De manera que el Autoencoder más sencillo se puede resumir como:

x x'

Otra función de pérdida que se suele intenta minimizar al hacer uso de *Autoencoders* es la función de *cross entropy*:

$$L = E\ x, x' = |x - x'|_p$$

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

Donde *M* es la dimensionalidad del input data x, que se suele asociar por ejemplo al número de píxeles en una imagen.

Sin embargo, este tipo de redes tan simples no pueden generar imágenes sin variaciones más allá de la pérdida de resolución típica de los Autoencoders. Para conseguir modificar la imagen de salida sustancialmente hacia una dirección deseada, nuestro modelo debe ser capaz de aprender la distribución de probabilidad de la muestra de entrada. Aquí es donde aparecen los *Variational Autoencoders*.

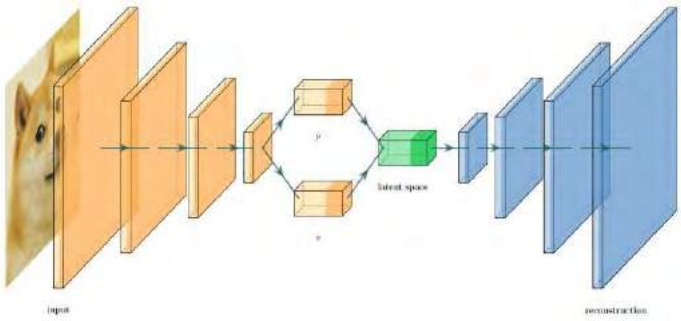

Fig. 12. Arquitectura típica de un Variational Autoencoder [12]

Este es uno de los enfoques más populares para aprender una distribución de datos compleja, como puede ocurrir con imágenes procesadas por Redes Neuronales de manera no supervisada. Es un modelo probabilístico enraizado en la inferencia bayesiana; es decir, el modelo tiene como meta aprender la distribución de probabilidad subyacente de los datos de entrenamiento para que pueda muestrear fácilmente nuevos datos en esa distribución aprendida. La idea sigue siendo la misma que la indicada para los *Autoencoders*: aprender una representación latente de baja dimensión de los datos de entrenamiento denominados variables latentes - variables que no se observan directamente, sino que se deducen a través de un modelo matemático - que asumimos que han generado nuestros datos de entrenamiento reales.

En este caso lo que se busca es modelar una entrada a nuestra red neuronal a una distribución. Llamemos a esta distribución p_θ, parametrizada por . La relación entre los datos de entrada x y el vector latentese puede definir como:

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

Si asumimos que se conoce el parámetro para esta distribución. Para poder generar una muestra que sea similar a una entrada real a la red, se siguen los siguientes pasos:

1. Primero se obtiene una muestra de $z^{(i)}$ desde una distribución prior $p_{\theta^*}(z)$
2. De manera que podamos generar desde una distribución condicional $p_{\theta^*}(x\ z = z^{\ i}\ )$ un valor $x^{\ i}$.

El parámetro óptimo es aquel que maximiza la probabilidad de generar muestras idénticas a los datos reales. Esto es:

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

Actualizamos la ecuación para demostrar mejor el proceso de generación de datos para involucrar al vector de codificación:

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

Desafortunadamente no es fácil caracteriza $p_\theta(x^i|z)$ para todos los posibles valores de *z*. Para facilitar una búsqueda más rápida, introducimos una nueva función de aproximación para dar salida dada una entrada *x*, *q (z|x)*, parametrizada por ø.

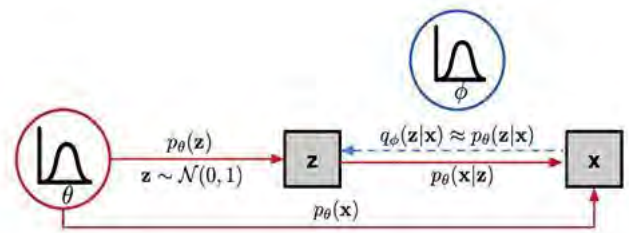

Fig. 13. Variational Autoencoder sampling. Las líneas continuas denotan la distribución generativa p_θ, las líneas discontinuas denotan la distribución para aproximar la posterior [14]

Ahora contamos con una estructura que se asemeja a un *Autoencoder*:

- La probabilidad condicional *pø (x|z)* define un modelo generativo, similar al *decoder*. A esta probabilidad se la conoce también como *decoder* probabilístico.
- La función de aproximación *qø (z|x)* se conoce como el *encoder* probabilístico.

El *qø (z|x)* posterior estimado debe estar muy cerca del real *pø (x|z)*. Podemos usar la divergencia de Kullback-Leibler [15] para cuantificar la diferencia entre estas dos distribuciones. La divergencia de KL *DKL (X|Y)* mide la cantidad de información que se pierde si la distribución Y se usa para representar X. Lo que se busca es minimizar DKL (*qø (z|x)|| pø (z|x))* (reversed KL) con respecto a ϕ. En este artículo sobre métodos variacionales bayesianos [X https://blog.evjang.com/2016/08/variational-bayes.html] se explica por qué se desea minimizar de esta manera en vez de DKL (*qø (z|x)|| pø (z|x))* (*forward* KL).

Si desarrollamos la divergencia KL:

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

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

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

Por lo que arreglando la ecuación:

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

El término de la izquierda de la ecuación es exactamente lo que se quiere maximizar durante el aprendizaje de la distribución verdadera de las entradas a la red: queremos maximizar la probabilidad *log p0 x* de generar datos reales y minimizar la diferencia entre lo real y lo estimado de DKL (*qø (z|x)|| pø (z|x))*.

Se puede definir la función de pérdida del *Variational Autoncoder*:

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

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

En los métodos Bayesianos Variacionales, esta función de pérdida es conocida como límite inferior variacional. Se denomina "límite inferior" porque la divergencia KL siempre es positiva y, por lo tanto, *-L VAE* es el límite inferior de *log p0 x*. Por ello, minimizando la función de pérdida, maximizamos el límite inferior de la probabilidad de generar muestras reales.

3.13. Reparametrización *trick*

Necesitamos generar muestras de *z ~ qø (z|xi)* para obtener nuevos datos de salida. Sin embargo, no podemos propagar hacia atrás el gradiente durante el entrenamiento si estamos generando estocásticamente muestras de esa distribución. Necesitamos un recurso que nos ayude; en el caso de los VAEs, se necesita muestrear desde una distribución gaussiana en el espacio latente de la red. El truco consiste en decir que el muestreo de *z ~ qø (z|xi) ~ N (z; ui, o2i I)* es equivalente al muestreo *e ~ N (o, I)* y definiendo *z = u + o · e*, en el que se refiere al producto por elementos de dos matrices.

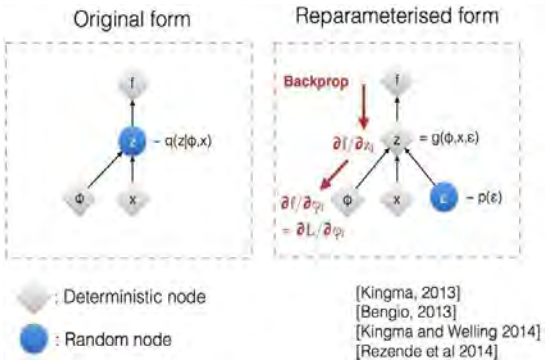

Fig. 14. Reparameterization trick para Variational Autoencoders [16]

En el caso que nos ocupa, procedemos a entrenar al modelo a aprender μ y de la distribución, lo que nos permite muestrear aleatoriamente la distribución $e \sim N(0, I)$.

Las GANs son el perfecto ejemplo de una red neuronal que constituye un modelo generativo haciendo uso del paradigma de aprendizaje no supervisado para entrenar dos modelos en paralelo en un juego de suma cero. Un aspecto clave de las GAN (y los modelos generativos en general) es cómo utilizan un recuento de parámetros que es significativamente menor de lo normal con respecto a la cantidad de datos con los que estamos capacitando a la red. La red se ve obligada a representar eficientemente los datos de entrenamiento, haciéndolos más efectivos en la generación de datos similares a los datos de entrenamiento.

La idea detrás del aprendizaje de este tipo de arquitecturas es simple pero muy ingeniosa. Se lleva a cabo un entrenamiento simultáneo de dos modelos: un modelo generativo G captura la distribución de los datos y genera muestras a partir de un modelo estadístico, mientras que un modelo discriminador D, que se comporta como un clasificador, estima la probabilidad de que una muestra provenga bien de los datos con los que se entrena el modelo o de que hay sido generado por G. El entrenamiento de este tipo de redes busca entrenar D para desenmascarar a G maximizando la probabilidad de que D esté equivocado.

Cuando el modelo discriminador D, que funciona como un clasificador binario, rechaza un ejemplo producido por el generador, el modelo generador aprende a refinar cada vez mejor la generación de nuevas muestras. ¿Cómo es capaz el generador G de producir ejemplos cada vez más cercanos a la realidad? En cada intento paso del entrenamiento, el discriminador D reporta al generador cómo de cerca se ha quedado de un ejemplo real.

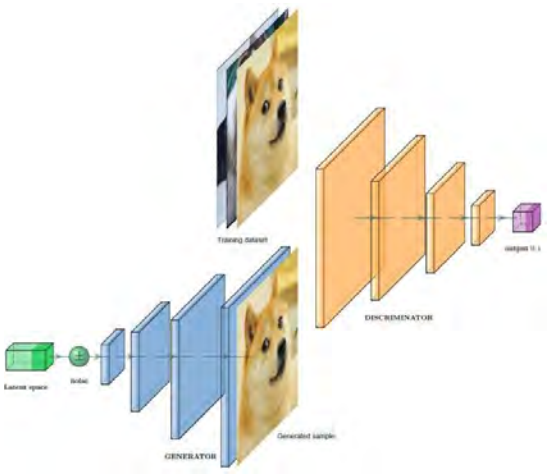

Fig. 16. Arquitectura típica de una GAN [12]

Por ejemplo, al modelar imágenes, el modelo discriminador suele ser una CNN estándar. El uso de una red neuronal secundaria como la red discriminadora permite a la GAN entrenar ambas redes en paralelo de manera no supervisada.

Estas redes discriminadoras toman imágenes como entrada y luego emiten una clasificación. El gradiente de la salida de la red discriminadora con respecto a los datos sintéticos de entrada indicará a la red generadora qué mínimos cambios realizar para generar datos sintéticos más realistas. La parte generativa de las GANs suele ser de naturaleza *deconvolucional*.

Durante el entrenamiento, se utiliza el algoritmo *backpropagation* en ambas redes para actualizar los pesos y biases del generador para generar imágenes de salida más realistas. Mediante el entrenamiento se busca actualizar los parámetros de la red de generación hasta el punto en el que el modelo discriminador es "engañado" por el realismo de las imágenes generadas en comparación con los datos de entrenamiento, siendo éste incapaz de clasificar correctamente si la imagen es real o falsa.

Para cada iteración se toma un conjunto de imágenes verdaderas y falsas y se actualizan los pesos de la red generadora y discriminadora a través de *backpropagation*. De acuerdo con *Goodfellow* et al. *Generative Adversarial Nets* [18], el problema reside en que este tipo de arquitecturas son difíciles de optimizar, ya que se intenta encontrar un punto de silla en vez de un mínimo de la función, por lo que el proceso de optimización resulta inestable.

3.14.Entrenamiento de una GAN

Si tenemos una imagen real x y definimos al ruido aleatorio generado como z:

- *D (x)* se define como la salida del discriminador cuando se le facilita una imagen real
- *G (z)* es una imagen obtenida del generador
- *D (G(z))* es la salida del discriminador a una imagen obtenida del generador

Idealmente, nuestro discriminador debería tener el siguiente comportamiento:

1. El discriminador deberá devolver un 1 siempre que se le facilite una imagen real, por lo que D(x) deberá maximizarse;

2. El mismo discriminador deberá asignar un 0 a toda imagen *G (z)*, por lo que *D (G(z))* deberá minimizarse. Por negación, *1 – D(G(z))* se deberá maximizar. Lo que queremos, por lo tanto, es maximizar 1 y 2, así que nuestra función objetivo es:

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

Para el generador G, la función objetivo debe corresponderse con la generación de imágenes con el valor de D(x) máximo, esto es, debemos minimizar V. Buscamos que el generador engañe al discriminador. Por lo tanto:

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

Construir y entrenar una red generativa con dos arquitecturas ligadas presenta un reto que hasta la aparición de este tipo no nos habíamos encontrado: tenemos dos funciones de pérdida, una del generador dirigida a crear mejores

imágenes y la otra perteneciente al discriminador focalizada en distinguir las imágenes generadas de las imágenes reales.

El entrenamos de ambos modelos es simultáneo, de manera que a medida que el discriminador mejora las imágenes reales de las imágenes generadas, el generador puede ajustar mejor sus pesos y sesgos para generar imágenes convincentes. En este caso nos encontramos a un discriminador y un generador participando en un juego de suma cero, donde las ganancias o pérdidas de uno se equilibran con las pérdidas o ganancias del otro.

Si combinamos ambas funciones objetivo:

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

Una vez definidas las funciones de pérdida, queda definir los optimizadores. El optimizador es independiente de cada red, y lo que buscamos es que cuando los pesos de una red están siendo actualizados, los de la otra se mantengan fijos. El optimizador minimizará la función de pérdida tanto para las dos funciones de pérdida del discriminador como para la del generador.

Puede ser difícil lograr la convergencia de las GAN converjan y, por lo general, suelen requerir de mucho tiempo de entrenamiento. Este no es el único problema: también nos encontramos con la dificultad de elegir la arquitectura óptima junto con los hiperparámetros correctos. No elegir la configuración correcta puede conducir a errores en la convergencia de la red.

Para las GANs, como comentábamos anteriormente, no existe una garantía de una única solución que *pdata (x) = pg*. Nos podemos encontrar con el conocido como *Failure to Improve*. Hay dos maneras en las que nuestro generador deje de mejorar, donde la mejora se toma con respecto a la calidad de las muestras que produce. En el primer caso, aunque exista una solución, la dinámica del algoritmo de entrenamiento de *Gradient Descent* evita que las Redes Neuronales alcancen sus valores de parámetros óptimos.

El otro caso se corresponde con que el gradiente a lo largo del cual debe entrenarse el generador, se reduce hasta el punto de que el generador no puede aprender de él. Esto se conoce como el *vanishing gradient problem* o el *saturation problem*. *Goodfellow* et al. (2014) afirma que este problema es causado por el discriminador, que es capaz de rechazar las muestras del generador con alta efectividad, de modo que el generador ve incapaz de continuar prendiendo. Por ello se debe evitar el sobre-entrenamiento del discriminador.

Otro modo bien conocido de error en el entrenamiento es el Mode Collapse. Es un problema que ocurre cuando el generador aprende a producir solo un rango limitado de muestras de la distribución de datos real. Lo hace asignando varios valores de entrada diferentesal mismo punto de salida. Este modo de colapso puede resolverse ajustando la arquitectura o facilitando al generador cómo de cerca se ha quedado de engañar al discriminador. De esa manera, se dirige el entrenamiento de manera más eficiente.

4. Faceswapping con VAEs

En la sección anterior de esta serie, se ha explicado el concepto de autocodificación y cómo las Redes Neuronales pueden utilizarlo para comprimir y descomprimir imágenes.

Entrenar una red neuronal significa optimizar sus pesos para lograr un objetivo específico. En el caso de un *Autoencoder* tradicional, el rendimiento de una red se mide según la forma en que reconstruye la imagen original a partir de su representación en el espacio latente. Es importante tener en cuenta que, si entrenamos dos *Autoencoders* por separado, serán incompatibles entre sí. Las caras latentes se basan en características específicas que cada red ha considerado significativas durante su proceso de entrenamiento.

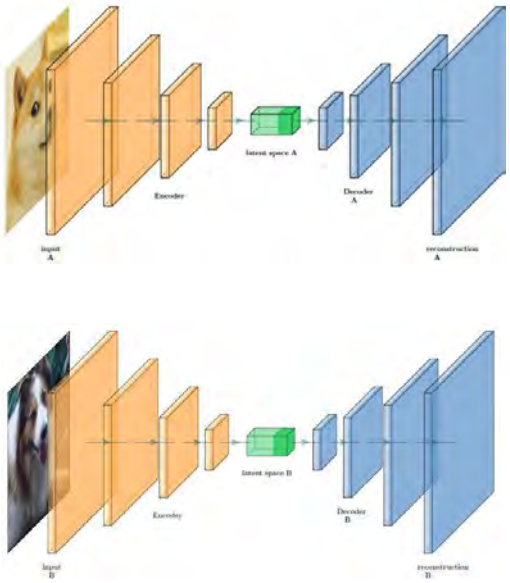

Fig. 17. Dos Autoencoders que comparten el mismo decoder aprendiendo diferentes distribuciones [12]

Lo que hace posible la tecnología de intercambio de caras haciendo uso de *Autoencoders* es encontrar una manera de forzar a ambas caras latentes a codificarse en las mismas características. Esto se consigue haciendo que ambas redes compartieran el mismo *encoder*, pero usando dos *decoders* diferentes. De esta manera, una vez acabado el entrenamiento de la red neuronal, se puede realizar lo siguiente durante el testeo de nuestro modelo:

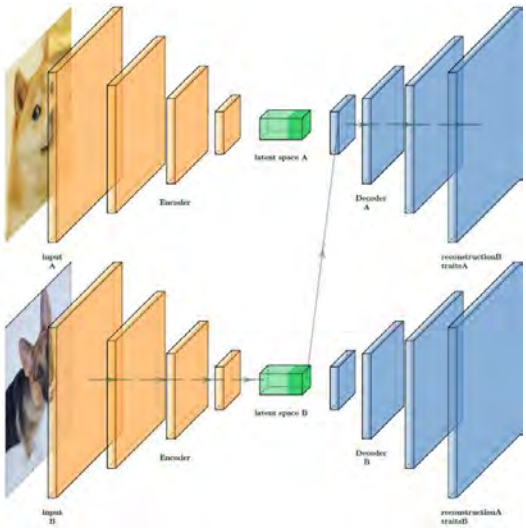

Fig. 18. Testeo de una red VAE dedicada a Faceswapping [12]

Si la red se ha generalizado lo suficiente como lo que hace una cara, el espacio latente representará expresiones faciales y orientaciones. Esto significa generar una cara para el Sujeto A con la misma expresión y orientación del Sujeto B y viceversa. Lo importante es que los dos temas utilizados en el entrenamiento comparten tantas similitudes como sea posible. Esto es para garantizar que el codificador compartido pueda generalizar funciones significativas que sean fáciles de transferir.

El problema, sin embargo, va más allá de las caras. La red neuronal no está diseñada específicamente para reconocer rostros. Las diferentes condiciones de iluminación y ángulos aumentan la complejidad del problema, ya que se requieren más pasos no solo para ajustar las caras, sino también para identificar las características relevantes. Si los cuadros utilizados para el proceso de entrenamiento muestran una alta variabilidad en las condiciones de iluminación, entonces el codificador se verá obligado a incluir información sobre ellos en la representación latente de la cara. Esto ocupa espacio que podría utilizarse para representar mejor las expresiones faciales, lo que reduce la calidad general. Esto se denomina *underfitting* e indica que el problema es demasiado complejo para ser capturado completamente con el modelo elegido.

Si bien eso es cierto, también tenemos que luchar contra el problema opuesto. Si todas las imágenes proporcionadas tienen el mismo aspecto, el proceso de entrenamiento podría no capturar su verdadera variabilidad. El resultado es que la red neuronal aprenderá a reproducir la misma imagen, en lugar de la misma cara. Este es un problema llamado *overfitting*.

Entonces, ¿qué es mejor? ¿Incluyendo imágenes que sean muy similares para simplificar el problema, o agregar tomas tomadas en diferentes condiciones de iluminación para generalizar mejor la apariencia del rostro? La respuesta es que depende. Es imposible saber cómo responderá la red neuronal simplemente diciendo que las imágenes se toman bajo la misma luz. En términos generales, sin embargo, cuanto más complejo es un problema, más difícil es resolverlo. A menos que sea un experto, debe esforzarse por ayudar a la red neuronal tanto como sea posible.

Si su objetivo es realizar un cambio de cara en un solo clip, no necesita una red neuronal que sea capaz de realizar un cambio de cara en cualquier condición. Solo necesitas uno que funcione solo una vez. La forma más fácil de lograrlo es reducir la complejidad del problema. Puedes comenzar seleccionando los videos correctos:

- Se deben usar videos filmados en condiciones de iluminación similares. La luz no solo afecta el color de una escena. También proyecta sombras en caras difíciles de emparejar. Esto se debe a que la reproducción de sombras realistas requiere una comprensión tridimensional de la geometría de la cara, que debe aprenderse a partir de imágenes bidimensionales.
- Se deben elegir sujetos con estructuras faciales similares. Esto va más allá del simple concepto de idea de "cara". El maquillaje, las gafas y la barba hacen que el problema sea mucho más difícil. Es particularmente difícil usar caras con características que vayan fuera de la cara misma (como barbas largas u vello oclusivo), ya que se recortarán.
- Por supuesto, el tono de piel debe ser similar. Una de las mayores desventajas del intercambio de caras es cómo las caras, una vez convertidas por la red neuronal, se vuelven a mezclar en el video. La forma en que funciona simplemente suaviza los bordes de la imagen de una manera bastante primitiva. Si las personas A y B tienen tonos de piel muy diferentes, será difícil fusionarlos sin problemas. Tenga en cuenta que ciertas implementaciones de cambio de cara ajustan los colores de la cara reconstruida para que coincida con el tono de piel original. Esta operación, desafortunadamente, se realiza en toda la imagen y afectará incluso las áreas que no necesitan ser corregidas (como los ojos y el vello facial).

5. *Faceswapping* con GANs

La capacidad de los modelos generativos para la generación de rostros ha quedado ampliamente demostrada. Sin embargo, con la aproximación basada en VAEs no podemos realizar esos cambios de cara en tiempo real. En esta sección se muestra cómo hacer uso de una *Generative Adversarial Network* para que un modelo aprenda puntos de referencia faciales detectados por una webcam y que estos se traduzcan en una cara haciendo uso de la librería pix2pix.

Lo primero que se debe realizar es obtener un conjunto de datos para poder comenzar el entrenamiento del modelo generativo. Los datos de entrenamiento se deben obtener sacando todos los frames posibles de cada video, así como

puntos clave del rostro de la víctima, denominados comúnmente como facial landmarks. Para esto último, se hizo uso del estimador dlib que puede detectar hasta 68 puntos de referencia (boca, cejas, ojos, etc.) tal y como se muestra en la Figura 19 en una cara junto con OpenCV para procesar el archivo de video. Los pasos a realizer y recomendaciones para una correcta extracción del rostro son los siguientes:

- Detectar los puntos de referencia faciales es un proceso de dos etapas. Primero, se usa una rutina de detección de rostros para detectar las caras y luego se aplica el estimador dlib a en la cara detectada.
- La estimación de la postura del rostro es una implementación derivada de la investigación.
- Es recomendable que el rostro de la víctima a suplantar no se mueva demasiado, así como garantizar que el vídeo los vídeos destinados a la obtención de los rostros de la víctima no contengan un fondo cambiante o muy complejo.

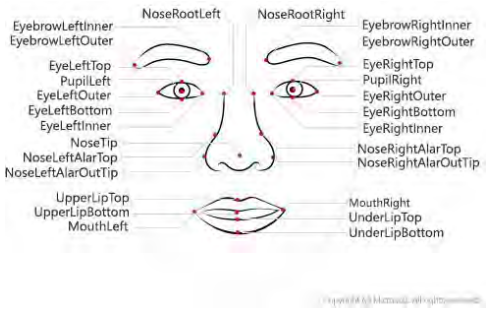

Fig. 19. Principales puntos de interés de los Face Landmarks [19]

Se hizo uso del repositorio face2face-demo [20], el cual toma toma los facial landmarks de todos los rostros extraídos de un vídeo para la generación de rostros haciendo uso de una Deep Convolutional Generative Adversarial Network (DCGAN).

Este repositorio, el cual se modificó siguiendo las instrucciones dejadas por Karolmajek para aumentar a la resolución de las imágenes con las que entrenar y testear, permite el entrenamiento de un modelo generativo haciendo uso del repositorio pix2pix-tensorflow (implementación original en Torch) [21]

El resultado del entrenamiento de este modelo, en un equipo con GPU, fue satisfactorio tras 3 días. Una vez entrenado el modelo y antes de proceder a la fase de testeo:

- Se reduce el modelo entrenado (se usa una imagen como tensor de input)
- Se procede a congelar el modelo a un único fichero

¿Qué es reducir y congelar un modelo en *Machine Learning*? Cuando queremos poner un modelo en producción, no necesitamos los metadatos adjuntos a nuestro *checkpoint*, sólo queremos que nuestro modelo y sus pesos estén bien empaquetados en un archivo. Esto facilita el almacenamiento, las versiones y las actualizaciones de sus diferentes modelos. De esta manera podremos ejecutar el testeo facilitando como entrada imágenes de webcam.

El uso de GANs para la suplantación de identidad es un campo en auge que, sin duda, en los siguientes meses a la publicación del presente *paper* dará mucho que hablar. Como hemos visto hasta ahora, tanto para la renderización de rostros como para el *Faceswapping* se necesita un gran número de muestras de fotografías de la víctima. Sin embargo, en muchos escenarios prácticos, no se cuenta con un *dataset* extenso, sino incluso con una sola imagen. En [21] se presenta un sistema que realiza un meta-aprendizaje prolongado en un gran conjunto de datos de videos, y después de eso puede encuadrar el aprendizaje de pocas o pocas personas de modelos de rostros neurales de personas nunca antes vistas como problemas de entrenamiento adverso con generadores y discriminadores de alta capacidad. El sistema puede inicializar los parámetros tanto del generador como del discriminador de una manera específica de la persona,

de modo que el entrenamiento puede basarse en solo unas pocas imágenes y hacerse rápidamente, a pesar de la necesidad de ajustar decenas de millones de parámetros. En esta publicación se demuestra que ese enfoque es capaz de modelar personas nuevas e incluso retratos, altamente realistas y personalizados [22].

6. Generación de audio

Con los recursos de *Faceswapping* en video investigados ya se contaría con la capacidad de generar un recurso multimedia no legítimo que podría ayudar a cualquier persona a suplantar a una víctima. Sin embargo, falta un importante recurso para completar el ataque: la generación de audio con el timbre y las inflexiones de la persona a suplantar.

Modelo personalizado de voz de la persona a suplantar. Para ello existen múltiples recursos que permite a un modelo generar muestras de audio a partir de texto escrito (Text-to-Speech) habiendo sido este modelo construido con muestras de audio de la persona a suplantar junto con sus transcripciones.

Dada la exposición digital a la que nos enfrentamos, no sólo resulta sencillo obtener vídeos de personas con cierta notoriedad que pueden ser foco de este tipo de ataques, sino que también obtener audio de sus voces es prácticamente inmediato gracias a los podcasts o a YouTube. En la presente investigación, se hizo uso de Neural Text-to-Speech (TTS) Microsoft: Deep Neural Networks para mejorar pronunciación y entonación. Este recurso de texto a voz utiliza Redes Neuronales profundas, lo cual le permite superar los límites de los sistemas tradicionales de texto a voz para hacer coincidir los patrones de acentuación y entonación en el lenguaje hablado, llamado prosodia, y para sintetizar las unidades del habla en una voz de computadora.

Los sistemas tradicionales de texto a voz dividen la prosodia en análisis lingüísticos separados y pasos de predicción acústica que se rigen por modelos independientes. Eso puede dar como resultado una síntesis de voz apagada y ruidosa. Este nuevo servicio hace la predicción de la prosodia y la síntesis de voz simultáneamente. El resultado es una voz más fluida y de sonido natural.

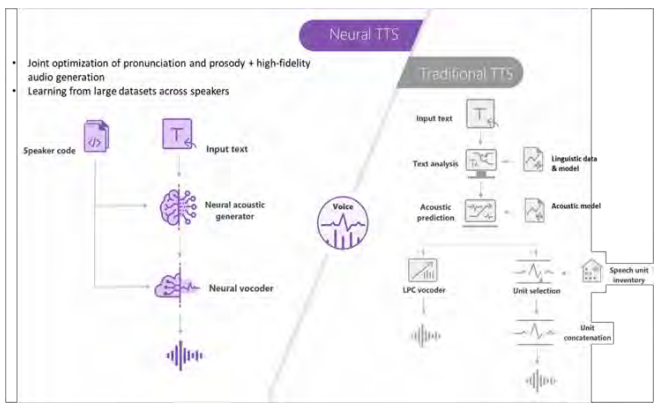

Fig. 20. Servicio Neural text-to-speech de Azure para síntesis de voz [23]

Esta personalización de voz le permite crear una voz reconocible y única para su marca. Para crear su fuente de voz personalizada, realice una grabación de estudio y cargue los scripts asociados como datos de entrenamiento. El servicio crea entonces un modelo de voz único sintonizado a su grabación. Puede utilizar esta fuente de voz personalizada para sintetizar el habla. Para poder hacer uso de Neural TTS, se debe contar con una cuenta de Azure.

Un conjunto de datos de entrenamiento de voz consiste en un conjunto de archivos de audio, junto con un archivo de texto que contiene las transcripciones de los archivos de audio. Es imprescindible comprobar las transcripciones de esos audios para un correcto entrenamiento. Para producir una buena fuente de voz, se deben realizar las grabaciones en una habitación tranquila con un micrófono de alta calidad, para obtener una buena relación señal-ruido. El volumen consistente, la velocidad de habla, el tono de habla y los gestos expresivos del habla son esenciales para construir una buena voz digital. Esto muchas veces no es posible, pues no se suele contar con la persona a suplantar para obtener buenas grabaciones del mismo. Normalmente, se recurre a muestras digitales de voz, que no necesariamente pueden ser de buena calidad o estar en la lengua materna del suplantado, lo que incluirá imperfecciones en la voz generada debido a SNR más bajas de lo recomendable o a scores de pronunciación más pequeñas de lo deseado.

Como primer paso, se procedió a aislar muestras de audio (en inglés) de la persona a suplantar. Al tiempo de escribir esta publicación, los únicos leguajes soportados eran el inglés y el chino.

1.Se descargan los audios (.wav) desde YouTube en inglés

2.En total se obtuvieron 314 *utterances* mono-canal de 30s cada uno, lo que equivale a 2.61 h de grabaciones.

Fig. 21. Dataset de utterances y transcripciones generado y subido al servicio de Custom Voice

3.Necesitábamos las transcripciones de los audios obtenidos. Para ello se usó Google Cloud Speech- to-Text API al tratarse de un recurso gratuito hasta una frecuencia de uso muy elevada; con esto se obtuvieron las transcripciones en formato .txt;

4.Subir audios y transcripciones para entrenar el algoritmo y obtener un modelo consolidado.

5.Una vez obtenido un modelo satisfactoriamente construido, procedimos al despliegue del mismo para testear.

El tiempo aproximado de entrenamiento de nuestra Font de Custom Voice fue de alrededor de 8 horas.

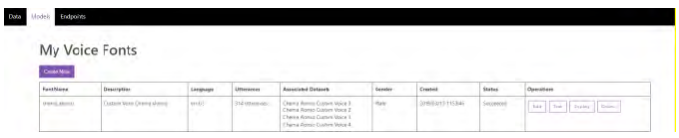

Fig. 22. Resultado tras el entrenamiento de nuestro modelo de Custom Voice

El resultado de la voz fue bueno a pesar de contar con un tercio del tiempo recomendado por Microsoft para tener una font de calidad. El timbre estaba bien logrado, pero entonación no dejaba de ser robótica. Se llegó a la conclusión de que un posprocesado de la voz se hacía imprescindible, siendo necesario tratar características de la voz como tono, timbre y vibrato, además de su inflexión. También se procedió a añadir un filtro telefónico y ruido de fondo para darle más credibilidad al texto generado.

7. Herramientas de protección

Como se ha visto hasta ahora, la potencia de los modelos generativos profundos ha mejorado significativamente la calidad y la eficiencia en la generación de videos de caras falsas de apariencia realista.

Hasta hace pocos años, resultaba muy sencillo distinguir un vídeo sobre el que se había aplicado la tecnología de *Faceswapping*, sin embargo, esto se está volviendo más complicado. La evolución de la calidad en generación de vídeos falsos es exponencial. Cada vez resulta más complicado distinguir un vídeo modificado de uno que es real. Si además le añadimos un soporte de voz para aumentar la probabilidad de que el *target* del ataque crea el engaño ¿Cómo protegernos de este tipo de ataques?

Al realizar la investigación, nos encontramos con la siguiente publicación [24]. En este trabajo, se describe un nuevo método para exponer videos de caras falsas generados con Redes Neuronales. El método se basa en la detección del parpadeo de los ojos en los videos, que es una señal fisiológica que no está bien presentada en los videos falsos sintetizados. Este método se prueba sobre los puntos de referencia de los conjuntos de datos de detección de parpadeo y también muestra un rendimiento prometedor en la detección de videos generados con *DeepFakes*.

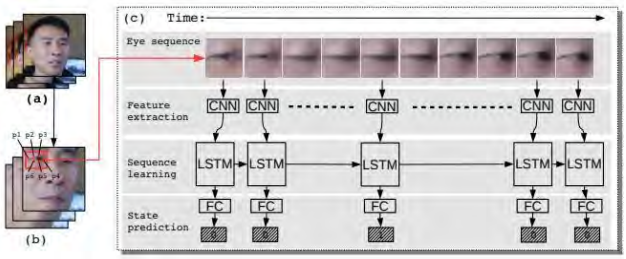

Fig. 23. Arquitectura recurrente convolucional empleada para la detección de parpadeos en [24]

El repositorio con el código está disponible en GitHub [26] y, aunque aún está en construcción, sirve como un ejemplo válido de aplicación de Redes Neuronales convolucionales recurrentes para el análisis de vídeos. En el equipo de Ideas Locas intentamos reproducir el funcionamiento de la herramienta, pero carecíamos del dataset original que nos permitiría calcular la probabilidad de que un determinado vídeo fuera falso o verdadero, por ello decidimos buscar una alternativa tal y como se indica a continuación.

Para una persona adulta sana, generalmente, entre cada parpadeo hay un intervalo de 2 a 10 segundos, pero las tasas reales varían según el individuo y la actividad que esté realizando. La velocidad media de parpadeo en reposo es de *17 parpadeos/min o 0.293 parpadeos/min* (durante la conversación, esta tasa aumenta a *26 parpadeos/min*, y disminuye a *4.5 parpadeos/segundo* mientras que la lectura de esta diferencia puede ser interesante en nuestro análisis, ya que muchos de los políticos que hablan probablemente estén leyendo cuando están siendo filmado). La duración de un parpadeo puede oscilar entre *0.1 – 0.4 segundos/parpadeo*. [27]

Por lo tanto, tomaremos como referencia de parpadeo normal una duración en el intervalo *0.1 – 0.4 s* y una tasa de parpadeo que oscile entre los *17-26 parpadeos/min*. Para desenmascarar vídeos falsos, decidimos crearnos un clasificador gaussiano con Scikit-Learn muy sencillo en base a dos características fácilmente extraíbles de un vídeo haciendo uso de las librerías dlib y opencv [28]:

- Velocidad de parpadeo [parpadeos/s]
- Duración del parpadeo [s]

Generamos muestras aleatorias para entrenamiento y testeo del modelo, lo que nos permitiría obtener un mapa de probabilidades de que el vídeo fuera etiquetado como Real o falso. Es este caso lo que buscábamos era llevar a cabo una regresión logística binaria sencilla (x para vídeo falso y para vídeo verdadero).

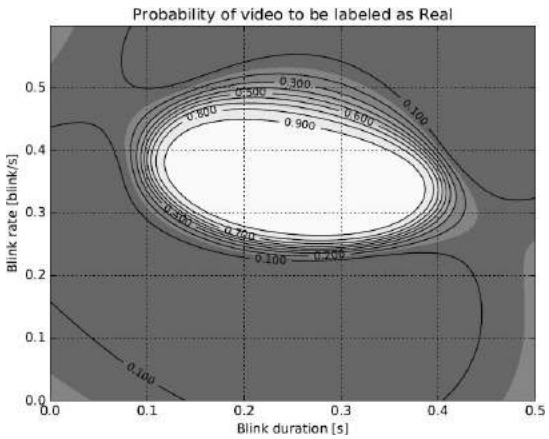

Fig. 24. Mapa de probabilidades de un clasificador gaussiano en base a la tasa y duración del pestañeo

Una vez que tuvimos nuestro modelo entrenado, sólo quedaba testearlo con los vídeos que queríamos desenmascarar. A pesar de la sencillez del modelo y de las pocas características extraídas, el modelo es capaz de discretizar si un vídeo es falso o no con bastante acierto. Esto pone de manifiesto que, si bien el modelo podría ser mejorable mediante la inclusión de características adicionales, con unas mínimas nociones de *Machine Learning* podríamos intentar protegernos contra este tipo de ataques.

Las redes sociales han cambiado drásticamente la forma en que las noticias son generadas, difundidas y consumidas por la sociedad, abriendo oportunidades imprevistas, pero también creando desafíos complejos. Las redes sociales, diarios digitales, si bien son fuentes muy importantes de información, permiten la generación de información altamente sesgada, así como facilitar la rápida difusión de la misma. Haciendo un uso fraudulento de las mismas, se pueden generar con facilidad campañas de información no legítima que pueden afectar la credibilidad de todo el ecosistema de noticias.

Una característica única de las noticias en las redes sociales es que cualquier persona puede registrarse como editor de noticias sin ningún tipo de restricción (por ejemplo, cualquiera puede crear una página de Facebook que diga ser un periódico o una organización de medios de comunicación). En consecuencia, muchos medios digitales tradicionales están emigrando cada vez más a las redes sociales. Junto con esta transición, no es sorprendente que haya una creciente preocupación acerca de los editores de noticias "falsos" que publican noticias "falsas", y que a menudo las difunden ampliamente utilizando seguidores "falsos". Como la extensa difusión de noticias falsas puede tener un impacto negativo grave en los individuos y en la sociedad, la falta de estrategias de verificación de hechos escalables es especialmente preocupante.

No es sorprendente que los esfuerzos de investigación recientes se dediquen no solo a comprender mejor este fenómeno, sino también a automatizar la detección de noticias falsas o *Fake News*. Si bien un enfoque totalmente automatizado para el problema de las noticias falsas puede ser bastante controvertido y aún está abierto para el debate. Una pregunta pertinente que nos podemos hace es: ¿Cuál es el rendimiento de la predicción de los enfoques y funciones actuales para la detección automática de noticias falsas?

La mayoría de los esfuerzos existentes en este espacio son trabajos simultáneos, que identifican patrones recurrentes en noticias falsas después de que ya se hayan difundido, o proponen nuevas características para los motores de clasificación.

Por lo tanto, es difícil evaluar el potencial que tienen los modelos supervisados entrenados a partir de las características propuestas en estudios recientes para detectar noticias falsas. Este artículo examinamos brevemente dos servicios que, haciendo uso de clasificadores de aprendizaje supervisado, nos permitirán clasificar la naturaleza del contenido digital que consumimos.

Cuando se comienzan a investigar las noticias falsas, podemos observar que hay muchas categorías diferentes en las que clasificar la información no verídica. Hay artículos que son descaradamente falsos, artículos que brindan un evento verídico pero que luego hacen algunas interpretaciones falsas, artículos que se pueden clasificar como pseudocientíficos, artículos que en realidad lo son sólo de opinión disfrazados de noticias, artículos satíricos y artículos que consisten principalmente en tuits y citas de otras personas. Es decir, pueden ser clasificados como "sátira ", "falso", "engañoso", "*clickbait*", etc.

La detección de noticias falsas es un problema de aprendizaje supervisado donde se aplican técnicas de procesamiento de lenguaje natural (NLP). En este caso nos encontramos con contenido web que puede obtenerse mediante web *scrapping*, y a cada noticia se la puede etiquetar en base a tantos criterios como se desee.

Una de las condiciones para que los clasificadores de noticias falsas logren un buen desempeño es tener suficientes datos etiquetados. Sin embargo, obtener etiquetas en las que confiar requiere mucho tiempo y trabajo de catalogación. Por lo tanto, los métodos semi-supervisados y no supervisados también se pueden utilizar para la agrupación de noticias en una primera aproximación exploratoria.

Fig. 25. Clasificación de texto sesgado como tarea dentro de paradigma supervisado

supervisado

Existen diversos servicios para la detección de *Fake News* basados en:

- Verificación de dominios de origen como fuente legítima o poco sesgada;
- Análisis de patrones en el contenido de la noticia;
- Búsqueda de *keywords* para mejorar la búsqueda, categorización y manejo de información;

Los modelos de clasificación más utilizados en la detección de noticias falsas son *Support Vector Machine* (SVM) y *Naive Bayes Classifier* (NBC). También se utilizan la regresión logística (LR) y los modelos basados árboles de decisiones.

El Deep Learning también tiene cabida. Muchos tipos de modelos de Redes Neuronales, como los perceptrones multicapa, también funcionan para la detección de noticias falsas. Las Redes Neuronales recurrentes (RNN) son muy populares en el procesamiento de lenguaje natural, especialmente en la memoria a largo plazo a largo plazo (LSTM), ya que pueden capturar dependencias a largo plazo.

En el departamento de ideas locas queríamos ver si era sencillo protegernos de este tipo de noticias, por lo que nos pusimos a investigar. Se encontraron dos servicios elegidos para el procesado y clasificación de noticia: Robinho y fakebox, tras lo cual generamos un breve script en Python que realizaba scrapping de una página web. Facilitada la url, el script obtenía todo el texto de la página. A continuación, se describen brevemente los dos servicios sobre los que nuestro programa trabaja:

- Robinho

El *Fake News Detector* de Robinho te permite detectar y señalar *Fake News, Clickbaits* y noticias Extremadamente Sesgadas.

Hay varias formas de usar el *Fake News Detector* [29]:

- Instala la extensión para Chrome o Firefox, esto verifica las noticias desde el feed de Twitter y Facebook.
- Habla directamente con Robinho en Telegram.
- Haciendo llamada a su JSON API, que es lo de que se ha hecho uso para el presente artículo.

- *Fakebox*

Fakebox analiza artículos de noticias para evaluar si es probable que sean noticias reales o no. Al observar una variedad de aspectos disponibles de un artículo (título, contenido y url) utilizando modelos de aprendizaje automático integrados y una *Fakebox* puede identificar con éxito noticias falsas con una precisión superior al 95%. *Fakebox* comprueba los siguientes aspectos de un artículo:

- Título. Los títulos pueden ser clickbait o sesgados
- Contenido: el contenido textual de un artículo se puede analizar para determinar si está escrito como una noticia real o no
- Nombre de dominio: algunos dominios son conocidos por alojar ciertos tipos de contenido, Fakebox conoce los sitios más populares al respecto.

El *script* generado, llamado fake.news.detector.py, se puede consultar en el siguiente repositorio [30].

Como puede observarse, el *script* es muy sencillo, no sólo en su implementación sino en su uso. Haciendo un GET y POST al siguiente diccionario de URLs:

```
url_main_dict= {

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

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

Facilitando los siguientes parámetros en cada llamada:

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

Se puede obtener el resultado de ambos servicios en formato JSON para acceder fácilmente a los datos.

8. Conclusiones

En esta publicación se ha intentado revisar el estado del arte y los casos de uso más populares de las tecnologías que, haciendo uso de Machine Learning y Deep Learning (modelos simples de ML, VAEs, GANs, Redes Neuronales Convolucionales y Recurrentes) pueden no sólo generar ataques de phising o Fake News, sino también protegernos de los mismos.

Las Fake News y las campañas de desinformación se han convertido en un verdadero problema en la actualidad. Citando a 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"

Cada vez es más difícil discernir qué información o recurso multimedia es real o no. Y conforme pasa el tiempo, las técnicas de generación mejoran su desempeño, por lo que su detección será cada vez más difícil con el paso del tiempo.

Como ha quedado demostrado, engañar a la gente puede salir 'muy barato'. Cualquier persona con una mínima base de informática y matemáticas puede adquirir un equipo medianamente potente que le permita entrenar modelos como los descritos en las secciones V, VI y VII. Ni siquiera necesita comprar un dispositivo; podría hacerse uso de servicios de Cloud (i.e., Google Cloud, Amazon Web Service, Microsoft Azure, etc) con los que crearse instancias virtuales que hagan uso de potentes GPUs o incluso TPUs (como las que facilita Google Cloud). El coste de estos servicios puede oscilar desde los pocos cientos hasta menos de 5000€ al mes, cantidad nada desdeñable que, sin embargo, podría resultar rentable si un pequeño porcentaje de las campañas de phising lanzadas tienen éxito.

Protegerse también es "barato". Con conocimientos básicos de *Machine Learning* podemos protegernos tanto contra texto sesgado o ilícito como contra vídeos modificados con fines espurios, como se ha mostrado en la sección 8.

Parece factible que la Inteligencia Artificial y la Ciberseguridad irán de la mano en el futuro. Aunque los límites de la IA todavía no se han vislumbrado, quedan demostradas las capacidades de ataque de modelos relativamente simples, haciendo uso de recursos que son de fácil acceso desde múltiples repositorios. Por ello, construir defensas que nos protejan contra estos ataques se convertirá en una necesidad.

Vivimos en una era de sobreinformación en la que podemos acceder a cualquier recurso informativo desde cualquier lugar del mundo a golpe de clic. La facilidad actual de generar y difundir contenido es una gran noticia para el saber general, pero también fomenta la generación de campañas de desinformación o engaño, cuyo volumen y sofisticación llega a ser tan elevado que hace imposible acotar su alcance en la sociedad. Por ello, las mejores herramientas para protegernos seguirán siendo el conocimiento, apostar por la educación y la concienciación, así como el espíritu crítico, cuestionando siempre la veracidad tanto de las fuentes como de la información que se consume.

9. Referencias

[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

Acerca de ElevenPaths

En ElevenPaths, la Unidad de Ciberseguridad de Telefónica, creemos en la idea de desafiar el estado actual de la seguridad, característica que debe estar siempre presente en la tecnología. Nos replanteamos continuamente la relación entre la seguridad y las personas con el objetivo de crear productos innovadores capaces de transformar el concepto de seguridad y de esta manera, ir un paso por delante de nuestros atacantes, cada vez más presentes en nuestra vida digital.

Más información

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