UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA

UNIÓN DE ASOCIACIONES DE INGENIEROS TÉCNICOS INDUSTRIALES Y GRADUADOS EN INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA (UAITIE)

“CONVOCATORIA 2022”

VII PREMIO NACIONAL DE INICIACIÓN A LA INVESTIGACIÓN TECNOLÓGICA

INTELIGENCIA ARTIFICIAL

AUTOR/ES:
José Manuel Moñino González, José Luis Sánchez Díaz, Jaime San Martín Pérez

BLOQUE TEMÁTICO:
Inteligencia Artificial

NIVEL EDUCATIVO:
Bachillerato

COORDINADOR:
Víctor Barbero Romero

ÍNDICE:

PALABRAS CLAVE:

Inteligencia Artificial, Machine Learning, Deep Learning, Redes Neuronales.

RESUMEN:

Este trabajo se centra en la descripción de la Inteligencia Artificial, sus diversos tipos y aplicaciones al mundo real, todo ello documentado con gráficas y figuras representativas que esbozan de forma sencilla algunos de los complejos conceptos que se tratan. No se busca una profundización excesiva, sino todo lo contrario, el trabajo adquiere un cierto tono divulgativo, y, por lo tanto, se busca la máxima comprensión posible apoyándonos en un lenguaje relativamente sencillo y elementos gráficos. Dentro de la Inteligencia Artificial, la investigación se centra en las redes neuronales o *Deep Learning* debido a su inmensa cantidad de aplicaciones y campos que trata, tras pasar por el *Machine Learning* o, Aprendizaje Automático, disciplina que engloba al *Deep Learning*. Finalmente, como parte práctica del trabajo se muestra un tutorial de instalación de red neuronal.

Objetivos:

El proyecto se centra en el contexto actual de la inteligencia artificial y su aplicación para resolver diferentes tareas complejas que los algoritmos previos propuestos no eran capaces de realizar con la misma precisión.

El objetivo fundamental se centra en recopilar una perspectiva completa que trate de responder la pregunta principal siguiente: ¿qué es la inteligencia artificial, para qué sirve y en que puede ayudarnos? El trabajo recopilará por tanto información detallada sobre todos estos aspectos buscando una presentación

formal de todas las bases de este campo de estudio.
Una segunda parte tratará de recolectar toda la información relativa a cómo funcionan estos algoritmos y cómo son sus implementaciones internas con énfasis en el proceso de aprendizaje que realizan sobre los datos (en este apartado no se profundizará en gran medida en las matemáticas involucradas, sino que se realizará un estudio sin llegar a ese nivel de detalle). Se centrará principalmente el trabajo en el campo del Aprendizaje profundo o Deep Learning del inglés presentando los tipos de redes neuronales que existen y qué propósito busca cada una de ellas.
Para concluir el trabajo se presentará un ejemplo sencillo de red neuronal implementada mediante un lenguaje de programación muy usado (Python) y además se presentará un tutorial completo para montar una red neuronal de las más complejas y actuales sin necesidad de tener unos conocimientos de programación

Metodología:

La metodología del proyecto seguirá una línea temporal creciente, es decir, no se puede pasar a comentar un apartado posterior sin haber descrito, detallado y entendido los previos.
Para recabar información se recurrirá a libros propuestos por los tutores, webs de confianza,
webs de consulta de artículos científicos e incluso noticias del tema pues es un campo que aparece continuamente en ellas.

Introducción:

La actualmente denominada Inteligencia Artificial se basa en el desarrollo de métodos y algoritmos que permiten a las computadoras comportarse de una forma inteligente. Este tipo de inteligencia es capaz de resolver un problema de una manera autónoma sin necesidad de interacción humana. Para desarrollar estos sistemas se emplea al ser humano y su cerebro como modelo de referencia, basándose en las conexiones neuronales con el fin de emular de forma matemática el comportamiento de un cerebro de manera programática.
Para determinar el nivel de inteligencia de un sistema o afirmar si es inteligente de algún modo se puede emplear el Test de Turing. Este test, desarrollado por el experto matemático del siglo XX y uno de los padres de la computación moderna, Alan Turing; se basa en la distinción de una inteligencia artificial con respecto a una persona. Si un juez externo no es capaz de diferenciar los resultados que provee una inteligencia artificial con los de un humano, significa que la inteligencia artificial es válida y está correctamente capacitada para desempeñar sus funciones. (*060031.pdf*, s. f.)
En algunos casos concretos podemos encontrar ejemplos prácticos donde la inteligencia artificial supera a la inteligencia humana ofreciéndonos unos mejores resultados en la resolución de problemas. Encontramos ejemplos muy comunes en los juegos clásicos de estrategia como pueden ser el cubo de Rubik: dónde los pequeños robots son capaces de resolverlo en menos de 1 segundo, mientras que el humano más rápido supera ampliamente el espectro de los 4 segundos basándose en un algoritmo supervisado capaz de aprender por sí mismo y distinguir los patrones que se acercan a la solución (Johnson, 2021); o en el ajedrez donde los ordenadores comenzaron a derrotar a grandes maestros en la disciplina desde 1980; pero no fue hasta 1997 cuando tras el encuentro entre el ordenador Deep Purple y el campeón del mundo del momento, Garry Kasparov.

Aún así, la inteligencia de la máquina fue muy cuestionada por los científicos de la época debido a que se construyó basándose en un algoritmo rígido, sin apenas margen de mejora, con un único propósito y con una inmensa base de datos de diversas jugadas realizadas por maestros mundiales (Hassabis, 2017). En el reconocimiento y distinción de objetos, las inteligencias artificiales vuelven a superar a los humanos. En 2015, la empresa de origen chino Baidu construyó el superordenador Minwa capaz de clasificar en miles de subgrupos las millones de imágenes que nos ofrece el set de datos de la ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) una competición encargada de evaluar y clasificar los algoritmos empleados en el reconocimiento (Wu et al., 2015).

1.2 Inversión en IA

Debido a la revolución tecnológica de la IA, en los últimos años las inversiones en inteligencia artificial han aumentado enormemente. En 2019 aumentó la inversión en un 635% llegando a 3500 millones de dólares de acuerdo con datos de Allianz Global Investors, el mercado de la IA debería crecer desde los 643,7 millones de dólares en 2016 a más de 37000 millones en 2025.

El país más importante en este entorno es Estados Unidos, dónde la Casa Blanca elabora un informe sobre el impacto de la IA en la industria (http://www.whitehouse.gov, 2018) y mantiene el liderazgo en lo que a Inteligencia Artificial se trata, representando más del 60% de la actividad mundial de citaciones de patentes de IA aunque está muy perseguido por China, que aumenta cada año sus publicaciones científicas en la materia de forma exponencial.

Debido a esta creciente demanda, en el ámbito académico han aumentado los doctorados especializados en Inteligencia Artificial y en machine learning, convirtiéndose en un 20% de los doctorados de las Ciencias de la Computación, lo que significa un aumento del 300% con respecto a 2004. El número de papers científicos también se ha incrementado, llegando a superar los 200.000 (según la base de datos Scopus en 2017).

Las empresas emergentes encargadas de desarrollar Inteligencias Artificiales se han disparado llegando a más de 600 empresas startup activas (valor triplicado en menos de diez años). Los beneficios para los emprendedores que generan también son enormes, superando los 2.000 millones de dólares (*2017-report.pdf*, s. f.). Este aumento en la tecnología no se ha detenido en el pasado año debido al Covid-19, sino qué al contrario, ha seguido aumentando como lo llevaba haciendo en años anteriores, las contrataciones han seguido el ascenso que llevaba sucediendo durante esta década debido a las inversiones masivas. La única excepción es EEUU, que ha decrecido en un 9% los empleos en comparación con 2019 Se buscan profesionales en Machine Learning, Deep Learning y en Natural Processing Language y sus salarios oscilan entre los 127.000 dólares anuales.

Durante la pandemia debido a las conferencias virtuales abiertas un gran número de personas ha tenido la oportunidad de adentrarse en el campo y observar su potencial. También es destacable la principal destinación de las Inteligencias Artificiales en 2020, principalmente al mercado de químicos y a la industria de la medicina encargada de realizar y obtener curas como la vacuna al virus con inversiones que ascienden a más de 13.000 millones de dólares (*2021-AI-Index-Report-_Chapter-3.pdf*, s. f.). Los países más implicados en esta tecnología

son Sudáfrica, Singapur, Cánada, Brazil, India, China, EEUU...

1.3 Aplicaciones de la IA en diferentes sectores

Debido a las infinitas capacidades que puede tener la Inteligencia Artificial esta es muy empleada en diferentes sectores, encargándose principalmente de labores relacionadas con la representación del conocimiento, buscando la resolución automática de problemas concretos debido a la dificultad de hacer a estos algoritmos capaces de resolver problemas diversos al mismo tiempo.

De este modo podemos encontrar diferentes y variados ejemplos reales como pueden ser robots encargados de explorar y desplazarse sobre diferentes planetas como los enviados a Marte por la NASA o diferentes empresas aeroespaciales que se encargan de tomar y enviar diferentes muestras de rocas de su corteza o estudiar las características de su atmósfera, la composición de gases, la textura del terreno, la presión. Estos robots funcionan gracias al entorno de trabajo del sistema autónomo científico OASIS (Onboard Autonomous Science Investigation System) (Estlin et al., 2007).

Otro sector muy beneficiado por la Inteligencia Artificial es el sector del entretenimiento. En él, diversas empresas tecnológicas como Google, Microsoft o Amazon emplean inteligencias artificiales para personalizar y ofrecer una mejor experiencia a sus usuarios. Otras como los servicios de streaming de películas de Netflix o HBO emplean algoritmos inteligentes para realizar recomendaciones personalizadas a cada usuario, ordenar las películas por categorías o realizar gráficos en base a las series o películas más vistas. Para ello se estudia las diferentes posibilidades que pueden hacer una serie exitosa, relaciones con series del mismo género, duración media, público destinado, fecha de emisión, diferencias entre los diferentes días de la semana. El algoritmo de Machine Learning ofrece al usuario las mejores recomendaciones tras analizar los diferentes datasets del servidor propio o encontrados en Internet.

En los videojuegos de ordenador se emplea la IA para la creación de PNJ (personajes no jugadores) que implementan un mayor dinamismo. Este caso es especialmente curioso ya que a medida que avanzan las generaciones los jugadores profesionales van exigiendo IA más inteligentes. Debido a esto surgen programas como NERO, un programa académico destinado a universitarios para desarrollar IA en diferentes videojuegos que mejore el comportamiento de los personajes o añaden dificultades más atractivas para el usuario individual. (Gold, 2005; Gunn et al., s. f.).

En domótica también tienen importancia las llamadas casas inteligentes con electrodomésticos dotados de inteligencia artificial para añadir una comodidad adecuada a los habitantes del hogar. Estas casas, entre otras muchas cosas, son capaces de aprender por sí solas, interactuar con el entorno y el usuario o resolver situaciones inesperadas o que no estaban calculadas (Bregman, s. f.).

Y por último, en medicina, donde tienen un gran papel realizando diagnósticos médicos de una precisión excelente como, por ejemplo, en el reconocimiento de tumores malignos. Se emplean algoritmos y robots de tecnologías asistivas en diversos programas innovadores de rehabilitación o en la curación de enfermedades como el párkinson. La Inteligencia Artificial y la medicina llevan conviviendo más de 50 años, en los cuáles se han producido diferentes etapas. Las primeras basadas en algoritmos de detección y clasificación de enfermedades; o de estudio y comprensión de enfermedades crónicas.

Pero actualmente principalmente se emplean redes neuronales en medicina molecular, haciendo estudios del ADN humano, sus variantes, sus mutaciones… (Altman, 1999)

2.-TIPOS DE INTELIGENCIA ARTIFICIAL

La inteligencia artificial es un campo extenso que cubre gran variedad de algoritmos y técnicas. Principalmente se divide en dos campos principales conocidos como Machine Learning y Deep Learning los cuales recogen actualmente todas las soluciones propuestas en este ámbito.

El Machine Learning se refiere a los algoritmos matemáticos que permiten aprender de los datos a partir de estos algoritmos entendibles, explicables y de un cierto modo interpretables. El principal problema que presentan son las diferentes limitaciones que tienen dependiendo del tipo de problema que se trate de resolver.

Además, los datos pueden llegar a suponer un incordio debido a que pueden tener limitaciones importantes cuando los datos de entrada presentan una gran complejidad en términos de relación.

El Deep Learning es un subgrupo del Machine Learning, un concepto más actual basado principalmente en redes neuronales complejas (algoritmos diseñados manualmente que requieren de comprobación de funcionamiento y observar si son capaces de aprender por sí mismos).Existen diferentes tipos de arquitecturas de redes neuronales las cuáles se escogen dependiendo del problema atendiendo a diferentes motivos. Pero si se decide observar qué es lo que ocurre en el interior de una red neuronal compleja estas se convierten en inexplicables. Estos algoritmos nos permiten trabajar con gran cantidad de datos omitiendo la "extracción de características" que requiere el "Machine Learning". Esta ventaja es la principal diferencia entre ambas.

-En el Machine Learning tradicional existe esa tarea previa de extracción de características la cual se basa en la aplicación de diferentes algoritmos para crear un vector de números que pueda representar de la manera más fiel posible al dato en cuestión. En Deep Learning esta etapa se omite pudiendo introducir los datos casi directamente aplicando muy pocos preprocesados iniciales a los datos, siendo así capaces de obtener estos vectores numéricos automáticamente durante el aprendizaje del algoritmo.

Otras cuestiones a tener en cuenta son que los algoritmos del Machine Learning funcionan bien con datos limitados y no demasiado complejos mientras que los del Deep Learning requieren de unas cantidades elevadas de datos para dotar al algoritmo de la inteligencia necesaria. El coste computacional es otra característica importante ya que un algoritmo de Machine Learning se puede entrenar en una máquina convencional mientras que uno de Deep Learning requiere recursos muy elevados como GPUs dedicadas.

2.1 Tipos de aprendizaje y problemas a resolver en inteligencia artificial

En la IA tenemos tres tipos de aprendizaje: un aprendizaje supervisado, en el que se conocen unos valores de entrada y se quieren obtener unos valores de salida que también se conocen previamente; un aprendizaje no supervisado, en el que únicamente se conocen los datos de entrada, es decir, los datos no están etiquetados, y entrenamos al algoritmo para que encuentre diferentes patrones o estructuras en los que agrupar los datos y

por último, un aprendizaje reforzado, que se basa en enseñar a la máquina las reglas del entorno donde se tiene que desenvolver y a partir de ellas establecer un sistema de recompensas y castigos que favorezca el aprendizaje.

Una vez vistos los tres tipos de aprendizaje principales se pasa ahora a presentar los diferentes tipos de problemas,que estos algoritmos pueden abordar y resolver de manera eficiente. No obstante, aunque no se haya presentado, dentro de estos tipos de aprendizaje existen otros que son al final combinaciones e ideas diferentes de los tres presentados como el aprendizaje semi-supervisado, dónde se tienen datos y etiquetados y otros no, o el aprendizaje por grafos dónde se crean estructuras de información conectadas a modo de grafo con diferentes nodos a los que se reparte la información, así como el aprendizaje por transferencia dónde se comparte un conocimiento previo de un algoritmo para resolver una tarea totalmente distinta. Los principales tipos de problemas a resolver se pueden encuadrar en cuatro categorías diferentes. Cada una de ellas tiene unas características diferentes y el resultado que se busca es muy diferente entre cada una de ellas.

Los de clasificación, en los cuáles queremos averiguar a dónde pertenecen los datos, un ejemplo típico puede ser la clasificación de imágenes y existen diversos tipos, la clasificación lineal, la bayesiana, pero la más generalizada es la binaria; El segundo tipo serían los problemas de regresión. En los anteriores casos, los grupos de datos eran discretos, pero cuando tenemos valores reales, o que no podemos establecer una relación entre ellos suelen ser resueltos mediante la regresión. Por ende, los problemas de regresión se basan en una función estimadora o regresora, De acuerdo con los modelos de regresión, las variables independientes predicen las variables dependientes (Roopa & Asha, 2019). Los análisis de regresión estiman los valores de “y” en un rango de valores de “x” (Seber & Lee, 2003). Los problemas resueltos pueden ser de dos tipos principalmente, para predicciones o pronósticos o para determinar la relación entre una variable dependiente y un set de datos adecuado..

En los problemas de Aprendizaje No Supervisado el objetivo suele ser agrupar los datos en base a características comunes, lo que conocemos como el **problema de agrupamiento o clustering**.

Otra capacidad interesante de los algoritmos de Clustering es, que no sólo son capaces de agrupar los datos, sino que también pueden hacer lo mismo con el flujo que generan y crear diversas clasificaciones de estos clusters (McGregor et al., 2004). Los problemas de clustering los podemos clasificar en dos tipos, Hard Clustering: En el cuál cada ápice de información permanece en un único clúster. Soft Clustering: En el cual cada punto puede pertenecer a varios clusters o agrupamientos.

Finalmente, otro tipo de problemas son los conocidos como **reducción de la dimensionalidad**. Se denomina reducción dimensional al proceso en el cual se reducen el número de variables de entrada con el objetivo de obtener un set de variables principales de menor complejidad.

3.-MACHINE LEARNING

3.1 Introducción al Machine Learning

Normalmente se asocia el origen del término Machine Learning al psicólogo Frank Rosenblatt de la Universidad de Cornell quien, basándose en el modelo del sistema nervioso humano y estableció un grupo encargado de crear una máquina capaz de reconocer las letras del

alfabeto. Esta máquina fue denominada perceptron (Brain & Rosenblatt, s. f.) por su creador usando señales analógicas y señales discretas e incluyendo un tercer elemento capaz de convertir las señales analógicas en señales discretas.

Centrándonos en el Machine Learning, este tipo de algoritmos empezaron a surgir y ser muy usados en la década de los 90. Las limitaciones existentes, en términos de capacidad computacional en cuanto a las redes neuronales artificiales se refiere, produjeron un cambio de paradigma y la aparición de numerosas soluciones matemáticas diferentes. Estos procesos han permitido crear gran cantidad de algoritmos capaces de resolver muchos de los problemas presentados con ciertas limitaciones. El primer problema clave es que estas soluciones presentan un paso previo de extracción de características manual, en contraste con Deep Learning que realiza la tarea de manera automática. Esto produce que sea necesario un gran trabajo previo de ingeniería de datos y matemáticas y estadística para encontrar una representación alternativa de los datos iniciales sobre la que poder aprender con los algoritmos de manera eficiente (Xin et al., 2018).

Este paso previo de extracción de características es clave para el uso efectivo del conocido Machine Learning.

3.2 Extracción de características

Existen numerosos tipos de datos y por ello diferentes formas de proceder a la hora de extraer información útil de cada uno de ellos para crear el vector final que se emplea para los algoritmos de aprendizaje.

Por ejemplo, comenzando con los datos en crudo, éstos pueden ser colecciones numéricas que ya presentan una estructura de vector. Cada número representará una característica diferente del conjunto seleccionado. Por ejemplo, si suponemos que tenemos datos de un coche, podríamos tener las características peso, número de puertas, caballos, precio, número de plazas, color… Cada una de estas características representa una información adicional y todas en conjunto forman nuestro vector de características buscado. La forma usual de tratar estos datos consiste en convertir toda la información no numérica (por ejemplo, color) a una representación numérica (por ejemplo, verde = 1, rojo = 2, negro = 3...).

Otra forma usual de trabajar es extraer diferentes parámetros básicos como la media, la desviación, los cuartiles entre otros parámetros que permiten conocer cómo estos se distribuyen y así adaptarlos, normalizarlos y realizar diferentes traslaciones a otros rangos de valores (mediante transformaciones simples).

Por ejemplo, algunas de estas técnicas pueden ser la extracción de luminosidad por la cual se extraen una serie de números que representan las diferentes intensidades de los píxeles en la imagen. Otra técnica muy empleada son los histograma de color ya que las imágenes tienen tres canales de color (RGB) y las diferentes partes y objetos en las imágenes pueden representarse adecuadamente gracias al color. Otras técnicas permiten extraer formas y texturas (algoritmos HOG, y LBP) que permiten representar lo que parece en la imagen de manera muy interesante. Otras técnicas más relacionadas con video pueden ser las características espacio temporales como el flujo óptico que permite ofrecer información del movimiento en la imagen.

3.3 Algoritmos de aprendizaje en Machine Learning

Los tipos de algoritmos más empleados en Machine Learning son los siguientes: de regresión, que se basan en predicciones y pronósticos (pueden generar confusión con los tipos de problemas pero realmente se considera antes un método de resolución antes que un tipo de problema); de reducción dimensional, que se basan en simplificar los datos (y encontramos además la misma posible confusión que en el caso anterior); basados en instancias, suelen crear una base de datos de datos de ejemplo y comparar los datos nuevos con la base de datos utilizando una medida de similitud para encontrar la mejor coincidencia y hacer una predicción; árboles de decisión, que toman como base valores reales de atributos en los datos para realizar diferentes bifurcaciones que se distribuyen en forma de árbol presentando resultados rápidos y precisos por lo que es uno de los tipos más empleados; bayesianos, en los que se aplica el teorema de Bayes para realizar clasificaciones o predicciones, redes neuronales muy básicas, como el perceptrón o el perceptrón multicapa, una función matemática capaz de realizar un mapeo simultáneo de los datos de entrada y de los datos de salida, reglas de asociación, que extraen reglas que explican mejor las relaciones observadas entre las variables en los datos y pueden descubrir asociaciones importantes y comercialmente útiles en grandes conjuntos de datos multidimensionales que pueden ser explotados por una organización, de agrupación, de regularización, que consisten en acoplar diferentes modelos de regresión constituyendo una modificación popular, potente y generalmente simple y de conjunto, formados por varios modelos más débiles que se entrenan de forma independiente y cuyas predicciones se combinan de alguna manera para hacer la predicción general.

4.-DEEP LEARNING

4.1 Introducción al Deep Learning

Los algoritmos basados en el Deep Learning permiten a los modelos computacionales crear algoritmos con múltiples capas y aprender la representación de los datos en diferentes niveles de extracción. La implementación de estos métodos ha revolucionado diferentes labores como el reconocimiento de objetos, de personas, de rostros o el descubrimiento de nuevos medicamentos de genomas en la industria del ADN. Estos modelos usando la retropropagación son capaces de adentrarse en las complejidades de los enormes sets de datos para cambiar sus parámetros internos y producir un mejor resultado final. El verdadero desafío para las inteligencias artificiales siempre ha sido resolver tareas que son sencillas de efectuar para las personas, pero difíciles de describir en el lenguaje formal. Algunos ejemplos son los mencionados anteriormente, procesos que realizamos de manera intuitiva, de forma automática (Sémery, 2018/2021).

Irónicamente, las tareas más complejas para realizar para un humano y que requieran gran intelecto como enormes cálculos matemáticos son las que los ordenadores realizan con mayor simpleza. La colocación jerárquica permite a los ordenadores aprender a resolver problemas complejos dividiéndolos en múltiples sencillos interrelacionados. Cada pequeño ápice de información que recibe el ordenador sobre el problema se llaman características. El modelo de aprendizaje del Deep Learning se basa en este aprendizaje de características o *Representation Learning* el cual engloba a todos los diferentes modelos encargados

de la identificación o correcta clasificación de características de un conjunto de datos en concreto. Estas representaciones elaboradas por el ordenador en ocasiones pueden ser mucho más útiles, precisas y exactas que las elaboradas manualmente por un individuo humano (Kolesnikov et al., 2020).

Cuando las características son clasificadas nuestro objetivo es separar los datos observados mediante los llamados factores de variación. En el contexto de las inteligencias artificiales la palabra factor se emplea para referirse a las fuentes de las influencias que nos ayudan a identificar las características. Pueden ser observables o no. También pueden ser constructos de la mente humana o conceptos y abstracciones que nos ayudan a enriquecer la clasificación. Por ejemplo, cuando analizamos un audio, los factores de variación corresponderían a la edad, sexo, acento del hablante o el vocabulario que emplea.

Por ejemplo, para la identificación de objetos o personas de una imagen, se pasan por diversas etapas, denominadas capas. En la primera capa o capa visible, la máquina recibe una entrada con un valor numérico de los píxeles representando un color. Después una serie de capas ocultas extraerán una serie de características de la imagen. Son nombradas capas ocultas debido a que sus valores no son dados íntegramente en la imagen, sino que los debe elaborar el algoritmo. Gracias a los valores de la primera etapa, por ejemplo, se pueden identificar fácilmente los bordes, después en la siguiente capa con los valores de las dos anteriores se intentará distinguir los contornos de los objetos o las esquinas. Por último, una tercera capa oculta nos ayudará a interpretar y analizar las diferentes partes del objeto y, dando un valor de salida con la identidad del objeto (si es un coche, una persona, un árbol).

Dos ideas principales son las que motivan a las redes neuronales a tener un fundamento biológico: el cerebro humano o animal como muestra de inteligencia y de un camino posible mediante inteligencia inversa de replicarlo en modelos computacionales o intentar entender el cerebro y los principios de la inteligencia para poder ser de utilidad en los algoritmos.

Otra de las esencias del Deep Learning es la retropropagación o Backpropagation. Esta técnica es la más empleada para el aprendizaje del algoritmo y consiste en volver atrás en el programa, cambiando los valores de peso de las neuronas para obtener el resultado correcto o que más se ajusta a la resolución de nuestro problema (LeCun et al., 2015) .

Los primeros ejemplos eran de forma lineal. The McCulloch-Pitts Neuron (Piccinini, 2004) fue un modelo temprano de las funciones del cerebro. Este modelo lineal podía identificar entre dos categorías comprobando si son positivas o negativas.

4.2 Funcionamiento general de las redes neuronales

Una red neuronal artificial consiste de simples unidades de procesamiento que se comunican enviando señales entre ellas sobre un gran número de conexiones llenas de “pesos”. Los principales rasgos de las redes neuronales, y, de forma interna en las neuronas en la son (Kröse et al., 1993) :

-Conjunto de unidades de procesamiento (neuronas o células).

-Una función o estado de activación para cada unidad que también equivale a su valor de salida.

-Conexiones entre las unidades. Generalmente cada conexión es definida por un peso “*w*” el cual determina el efecto en el cual la señal tiene influencia.

-Una regla de propagación que determina el valor de entrada efectivo de una unidad sobre sus valores externos de entrada.
-Una función de activación la cual determina un nuevo nivel de activación basado en el valor de entrada efectivo y su activación actual.
-Un valor externo llamado *bias* que es capaz de alterar el valor de salida de cada unidad y poder cambiar el sentido de la activación.
-La regla de aprendizaje o un método para recopilar información.
-Un entorno en el cual el sistema opera ofreciendo valores de entrada y, si es necesario, señales de error.

Cada unidad de procesamiento ejecuta una labor relativamente simple, recibir valores de entrada de sus vecinos o fuentes externas y usar esto para computar un valor de salida propagable a otras neuronas cercanas. El sistema es completamente paralelo, en el sentido de que varias unidades son capaces de transportar sus cálculos computacionales al mismo tiempo.

A parte de esta tarea, también realizan una segunda, para ajustar los pesos de las neuronas, estos serán los valores que irán cambiando durante el aprendizaje para acercarse a los óptimos que resuelvan el problema.

Durante las operaciones, las unidades pueden actualizarse de forma sincronizada o de forma no sincronizada. Si es de forma sincronizada, las neuronas proceden a tener activaciones simultáneas y si es de forma no sincronizada, normalmente una única neurona procede a activarse en un tiempo determinado "*t*" lo cual puede ser ventajoso dependiendo del tipo de problema al que nos enfrentemos. El valor total es simplemente la suma de las neuronas separadas y de sus conexiones mediante un *bias* si se da el caso. Si la contribución es positiva se considera excitación y si es negativa se considera inhibición. En algunos casos se emplean complejas reglas de propagación para combinar entradas haciendo distinción en las contribuciones de excitación y de inhibición.

También es necesaria la existencia de reglas que manipulen el efecto total de la activación de cada célula. Se necesita una función en la cual se toman el total de valores de entrada y la activación actual para producir un nuevo valor de activación.

Descrito el funcionamiento de la unidad básica de procesamiento, el perceptrón, existen diferentes topologías de red, es decir, los patrones de conexiones entre las unidades y la propagación de los datos. Teniendo en cuenta los patrones de conexiones podemos distinguir en:

-Feed-fordward networks: En las cuáles el flujo de datos de entrada y de salida es estrictamente de forma prealimentada, es decir, puede extenderse por múltiples capas, pero no puede tener conexiones extendiéndose entre la salida hacia la entrada de la capa para volver a entrar en la misma capa o en capas previas.

-Redes recurrentes que tienen conexiones feedback: De forma contraria a las redes anteriores, las propiedades dinámicas de la red son importantes. En algunos casos los valores de activación de las unidades son relegados a un proceso de relajación y se espera que la red evolucione a un estado estable en el cuál las activaciones no cambien más.

Para otras aplicaciones, es de importancia los valores de salida de las neuronas así como el comportamiento dinámico que de la salida de la red (Pearlmutter, 1991).

Una vez seleccionada la tipología para el problema, se pasa a crear la arquitectura de red. Esto dependerá de gran cantidad de factores y el diseñador será el que emplee tantos bloques como considere necesarios y unidades de información en cada capa. Creada la arquitectura final de la red el siguiente paso es entrenarla para

que aprenda para ello existe un algoritmo conocido como el, backpropagation que se comentará en párrafos posteriores.
Una red neuronal es configurada para obtener unos valores deseados de salida a partir de unos de entrada. Varios métodos convencionales se basan en endurecer las conexiones para obtener el resultado esperado. Una manera es configurar los pesos de forma *a priori*. Otra manera es alimentar a la red neuronal para que aprenda basándose en patrones y dejándola manipular los pesos en base a sus propias reglas formuladas.
Una simple red es capaz de representar la relación lineal presente entre el valor de la unidad de salida y el valor de las unidades de entrada. Del umbral de los valores de salida, es posible construir un clasificador. En espacios de altas dimensiones la red representa un hiperplano que además puede ser representado de diferentes unidades de salida definidas.

Suponiendo que disponemos de un hiperplano y queremos encajarlo de la mejor forma posible a un set de entrenamiento basado en valores de entrada y unos valores de salida deseados; cada valor de salida diferirá del esperado. La regla delta, emplea la función de error o coste que surge de las diferencias de los pesos de los valores de salida. De esta forma, modifica los pesos apropiadamente para los objetivos de salida de cualquier polaridad y para unidades binarias de entrada y salida.
El aprendizaje por retropropagación puede considerarse una generalización de la regla delta para las funciones no lineales y las redes multicapa adaptando únicamente unos pocos parámetros de error.
Cuando ya tenemos estructurado el patrón de aprendizaje y las funciones de activación comienzan a propagarse es muy probable que termine con un error en alguna de las unidades de salida. Para cambiar este error, la manera más simple es cambiar las conexiones apoyándonos en la regla delta. Pero esto no es suficiente, ya que las unidades ocultas nunca han cambiado. Podríamos intentar repetir el proceso con la regla delta, pero faltan algunos parámetros de error y por eso se recurre a la regla de la cadena en la cual se distribuye el error por las diferentes capas ocultas. El resultado final es la existencia de valores deltas en las unidades ocultas los cuáles ya si son manipulables y por lo tanto se puede modificar el peso de las neuronas para completar la retropropagación y así, el aprendizaje del sistema. Esta regla es muy simple y puede empezarse así:
1.-Se empieza con pesos aleatorios para las conexiones
2.-Se selecciona un vector de entrada x del set de entrenamiento
3.-Si el perceptrón da una respuesta de error modifica las conexiones wi de acuerdo a $\Delta wi = d(x)\, xi$; siendo d (x) el valor deseado de salida
4.-Volver al paso 2

4.3 Tipos de redes neuronales

Dependiendo del tipo de datos se han creado diferentes estructuras de redes neuronales que son óptimas para trabajar con cada tipo de datos. Las primeras arquitecturas que aparecieron fueron las conocidas como redes neuronales artificiales o ANN del inglés y las profundas donde existen un gran número de capas ocultas. Estas toman como entrada un vector de datos y producen una salida dependiendo del problema a resolver de los vistos en el apartado de tipos de problemas en inteligencia artificial. El segundo tipo son las redes neuronales convolucionales y que son aptas para trabajar con datos de entrada en forma de matriz como las imágenes dónde se busca encontrar relaciones

espaciales entre los datos. Finalmente, un tercer tipo son las redes neuronales recurrentes, las cuales son válidas para datos con dependencia temporal o jerarquía en el orden como es el audio o el texto. En base a estos tres tipos se pueden crear diferentes arquitecturas que han ido tomando nombres diferentes como los autoencoders, las redes generativas o los modelos de atención.

Tras describir la arquitectura, el aprendizaje es otro componente esencial de la red neuronal. Se pueden crear funciones de error haciendo el promedio de las raíces de los valores entre salida y entrada.

Una red neuronal jamás podrá aprender algo que no esté presente en su conjunto de datos de entrenamiento. El tamaño de un set de entrenamiento debe de ser lo suficientemente grande para que la red neuronal sea capaz de memorizar y aprender las diferentes característicaspero no debe superar un límite, en el cual se comiencen a desperdiciar recursos eliminando “ruido” o datos innecesarios para nuestro objetivo. Por lo tanto, una buena selección de los datos de entrenamiento es esencial para una correcta resolución del problema.

Una de las redes neuronales más populares es la **red neuronal convolucional** (CNN). Recibe el nombre de la operación matemática lineal entre matrices denominada convolución. Tiene múltiples capas incluyendo: la capa convolucional, la capa no lineal, la capa de agrupación y la capa completamente conectada. A medida que los datos son procesados por las diferentes capas, se va reduciendo el tamaño de la información.

La capa completamente conectada y la capa convolucional tienen parámetros, pero la capa de agrupación y la capa no lineal no los tienen. Las CNN tienen un excelente nivel de actuación en tareas de Machine Learning; especialmente cuando sus aplicaciones están relacionadas con las imágenes, visión computacional y el procesamiento del lenguaje natural. La principal ventaja de las redes convolucionales es que tienen un número más reducido de capas que las ANN y por lo tanto, se pueden crear modelos más grandes capaces de trabajar con cantidades enormes de datos. En los problemas de las redes convolucionales, destacan que las características no deben ser estrictamente dependientes. Por ejemplo, si tenemos un sistema de reconocimiento facial, no se necesita centrarse en apartados innecesarios como la localización. Por último, cuando el sistema entra en funcionamiento, también se pueden producir la extracción de características abstractas en las capas ocultas.

En las redes neuronales artificiales se emplean técnicas para el reconocimiento de patrones estructurales que superan a los tradicionales. En estas técnicas, los atributos de los datos de muestra son presentados a la red neuronal al mismo tiempo. Después de un entrenamiento exitoso, la red neuronal debe ser capaz de categorizar las características que se encuentran en el conjunto de entrenamiento.

Una **red neuronal recurrente o RNN** es un tipo red que posee una retroalimentación similar a la retroalimentación de los circuitos electrónicos. En las redes neuronales recurrentes se implementa un ciclo, conectando por ejemplo las capas ocultas con ella misma por pesos, conectar capas ocultas con las capas de entrada, o incluso conectar todas las unidades entre ambas, pero, sin embargo, la complejidad de estas capas nunca aumenta. sino que al contrario, puede decrecer el tamaño de la red para resolver el mismo problema. Existen redes más complejas como, de forma independiente, las redes de Almeida (Almeida, 1990) y Pineda

(Pineda, 1987) que descubrieron errores en la retropropagación que pueden solucionarse con un caso especial de un aprendizaje de gradiente general empleado para entrenar redes con atractores. Sin embargo, cuando no se espera que la red alcance un punto concreto se puede usar una retropropagación.

6.-Tutorial para usar algoritmos de IA

En esta sección se va a presentar como preparar un ordenador para utilizar algoritmos de inteligencia artificial de los más actuales de manera sencilla.

Los pasos principales que se van a seguir son los siguientes:

-Instalación del entorno de programación de código: Python

-Instalación de herramientas para gestión de descargar/modificación de código: Github

-Instalación de la librería base Python para desarrollo de algoritmos de inteligencia artificial:Pytorch

-Instalación de los requerimientos básicos que el algoritmo que vamos a utilizar necesita para ser usado. Esto implica la instalación de librerías de código (en este caso Python) adicionales que permiten la ejecución correcta de diferentes partes del código básico que se va a emplear. En ocasiones hacen falta otro tipo de programas o instalaciones adicionales, pero, los repositorios suelen venir acompañados de un tutorial para instalar todo lo necesario de cara a hacer funcionar los algoritmos sin problemas adicionales.

Para instalar el entorno de programación se va a hacer uso de la suite de programación conocida como Anaconda. Esta suite presenta un conjunto de herramientas que incluyen la instalación completa del sistema de programación Python, así como otra serie de ayudas a la programación: un entorno eficiente y simple para la escritura de código y su ejecución u otras herramientas para testeo y análisis de código.

Este tutorial se va a basar en Windows puesto que es el sistema operativo más extendido entre todas las personas y de cara mostrar que cualquier persona con un ordenador puede utilizar y montar estos algoritmos sin complicaciones adicionales.

Para comenzar por tanto se va a instalar el entorno de programación Python en el ordenador con sistema operativo Windows. Para ello se va a hacer uso de la suite de programación conocida como Anaconda. Esta suite incluye el sistema de programación Python como numerosas herramientas para desarrollo de código testeo entre otras. Este programa se puede descargar desde el siguiente enlace:

https://www.anaconda.com/products/individual

Una vez se accede al enlace, se presiona el botón de descarga y esto almacena en el ordenador el fichero de instalación de la suite de Anaconda. Una vez descargado, este se ejecuta y pedirá una serie de información para la instalación dónde sólo hay una parte importante en la que fijarse si esta no viene previamente marcada.

Una vez pasadas varias de las opciones de instalación y dejarlas por defecto sin modificar nada.

Por defecto viene “Desmarcada la primera opción”. Marca esta opción antes de continuar para poder utilizar los comandos de anaconda desde el terminal de Windows cosa que presentaremos más adelante.

Tras esto, continúa aceptando las configuraciones por defecto que sigan apareciendo durante la instalación y tras esto el programa comenzará a instalarse en el ordenador. Una vez haya finalizado tendrás en tu ordenador instalada la suite de programación Anaconda con el lenguaje de programación Python en su

última versión. Para comprobar que todo es correcto se va a proceder de la siguiente manera:
-En el buscador de windows escribe “cmd” . Aparecerá un programa llamado “Símbolo del sistema” en español. Abre ese programa.
-Una vez abierto aparecerá una pantalla negra como la que se muestra en la siguiente figura dónde se puede escribir. Vamos a escribir “conda” que es el comando básico para controlar anaconda de manera programativa sin entrar a la interfaz gráfica de programa. Si esto funciona debería mostrar una información sobre cómo funciona este comando, si no, debería aparecer algo como este programa no se reconoce en el sistema operativo.

La segunda parte va a consistir en instalar la herramienta base que sólo usaremos en este tutorial para la descarga del código público que queramos utilizar. en el siguiente enlace está disponible la descarga: https://git-scm.com/download/win
En principio, aceptando todo por defecto debería instalarlo todo de manera correcta (sino hay una opción a marcar que menciona algo sobre incluir en “bash”) y de la manera igual que la anterior tras la instalación, abriendo un terminal y escribiendo “git”, al presionar “enter” debería presentarse cierta información dando a entender que este programa está instalado adecuadamente en el sistema y preparado para usar mediante comandos desde terminal.
El siguiente paso va a ser instalar la librería Pytorch, pero podrían ser otras como Tensorflow o Keras (desarrolladas por Google) entre otras proporcionadas por otras grandes compañías de software. Para instalar Pytorch sólo debemos entrar en su página web (https://pytorch.org/get-started/locally/) y elegir el sistema operativo que deseamos Esto nos generará una línea de código que introduciremos en un terminal para proceder a la instalación. En la siguiente imagen se puede ver la elección de estos parámetros para obtener la línea de código necesaria:

```
conda install pytorch torchvision torchaudio cpuonly -c pytorch
```

Dado que no vamos a tratar con problemas de procesado como el uso de GPUs o otros dispositivos hardware que ayuden al procesado de esta información, se va a realizar la instalación para el uso de esta librería con funcionamiento CPU, es decir usando directamente el procesador del ordenador. Para ello la línea de código previa aparece en base a la siguiente elección de parámetros. Para proceder, se copia esta línea en un terminal de Windows y debería preguntar tras unos segundos si se quiere instalar esta librería preguntado por un sí o no. Para aceptar se escribe “y” que significa “yes” y se presiona “enter” lo que dará comienzo a la instalación.
Vamos a instalar uno de los algoritmos más famosos para detección de objetos conocido comoFaster-RCNN desarrollado por facebook y que es capaz, con la versión base, de detectar 80 objetos diferentes para los que ha sido entrenado.
El siguiente paso es instalar una librería adicional necesaria en Windows para el uso adecuado de estos algoritmos, por lo tanto, en un terminal introducimos la siguiente línea de código, aceptamos la instalación escribiendo “y” y presionando “enter”.

```
conda install pywin32
```

Una vez hecho esto podemos comenzar con la descarga e instalación del algoritmo a usar. Para ello seguiremos el tutorial proporcionado por desarrolladores de la comunidad dónde se proporcionan los pasos para instalar el algoritmo de detección de objetos que buscamos: https://dgmaxime.medium.com/how-to-

easily-install-detectron2-on-windows-10-39186139101c

Apoyándonos en este enlace, seguiremos los pasos para la descarga e instalación. Para ello abrimos un terminal de nuevo y seguimos los siguientes pasos:

- `pip install cython`
- `pip install "git+https://github.com/philferriere/cocoapi.git#egg=pycocotools&subdirectory=PythonAPI"`
- `git clone https://github.com/DGMaxime/detectron2-windows.git`
- `cd detectron2-windows`
- `pip install -e .`
- `pip install opencv-python`
- `python tests/test_windows_install.py`
- Si este proceso ha sido exitoso el resultado del último comando debería mostrar el siguiente resultado.

Si hasta este punto todo ha funcionado de manera adecuada, ya se tiene entonces el algoritmo funcionando en tu propio ordenador. ¿Qué podemos hacer entonces? Vamos a introducir nuestras propias imágenes para ver qué sucede. Para ello hay que modificar unas líneas de código muy simples pues la imagen previa se obtiene a través de la web y ahora vamos a cargar nuestras propias imágenes en el ordenador. Tras todo este proceso se habrá descargado en nuestra carpeta personal el código completo del algoritmo. Si entramos en la carpeta y dentro de la misma en una carpeta adicional llamada "tests" encontraremos un archivo llamado "test_windows_install.py". Abrimos este archivo y veremos lo siguiente:

Este es el código básico Python que ejecuta el algoritmo que hemos descargado en nuestro ordenador. Vamos a añadir una línea adicional para poder cargar nuestras imágenes propias puesto que por defecto en este trozo de código se descarga la imagen previa desde internet y es la que se usa para la predicción. Por tanto, para ello, sin entrar en ningún aspecto de programación, escribimos la siguiente línea tras la línea 14 en esta imagen previa:

`cv2.imread("nombre de la imagen")`

Para usar nuestra propia imagen tenemos que hacer dos cosas, la primera es poner en esa línea de código el nombre de nuestra imagen, por ejemplo, si tenemos una imagen que se llama "perro.jpg" pondremos estos dentro de esa línea de código. La segunda es poner las imágenes que queremos en el directorio correcto que sería la carpeta principal que hemos descargado, es decir, la carpeta en nuestra carpeta personal llamada "detectron2-windows". Aquí copiamos todas las imágenes que queramos y en el código introducimos los nombres para usar unas u otras. Por ejemplo, vamos copiar una imagen llamada "perro.jpg" y vamos a realizar la detección de objetos en esta imagen entonces primero copiamos la imagen a la carpeta.

Una vez copiada modificamos el código con el nombre de la imagen.

Finalmente lanzamos de nuevo en un terminal la última línea de código que llama al algoritmo para ser ejecutado:

`python tests/test_windows_install.py`

El resultado en la imagen que hemos seleccionado de un perro es el siguiente, dónde podemos observar que nos aparece un recuadro alrededor del perro que el algoritmo ha predicho y estima que el perro se encuentra en esa parte de la imagen y además no dice una información adicional, en la esquina del recuadro podemos leer "dog", es decir el algoritmo sabe que lo que hay en la imagen es un perro, y por otro lado vemos un "100%" que representa la probabilidad con la que el algoritmo cree que eso es un perro. En este caso el algoritmo está totalmente

seguro de que eso es un perro, por lo que empieza a dar a entender que estos algoritmos empiezan a equipararse a los humanos en ciertos aspectos.

Vamos a introducir un ejemplo adicional para ver qué sucede con otros objetos. Como se ha comentado este algoritmo es capaz de detectar 80 objetos que son los que aparecen en el dataset conocido como COCO (*COCO - Common Objects in Context*, s. f.) y de dónde se puede encontrar toda la información relativa en el siguiente link (https://cocodataset.org/#homeç).

Hemos copiado una nueva imagen llamada “perros_personas.jpg” dentro de la carpeta del proyecto.

Podemos observar cómo los perros y las personas han sido detectados. Si nos fijamos más detalladamente hay un perro al fondo que sostiene una persona el cual no ha sido detectado y esto puede ser debido a que al no aparecer completamente y estar parte del mismo ocluido el algoritmo no es capaz de encontrarlo eficientemente. Si nos vamos a otro detalle interesante es que en la persona que se encuentra más a la derecha se detecta un objeto adicional que es una bolsa o mochila que lleva ésta.

De esta manera podemos usar este algoritmo para introducir las imágenes que queramos de cara a detectar diferentes objetos en las mismas. De nuevo, hay que recalcar que este algoritmo es capaz de detectar 80 objetos y son los presentes en el conjunto de datos con los que ha sido entrenado. Estos objetos se pueden consultar en el enlace previamente presentado, pero el algoritmo no es capaz de detectar otros objetos diferentes de esos presentes en ese conjunto de datos. Aquí aparece una limitación importante de la inteligencia artificial y es que los algoritmos son limitados y se centran en tareas concretas y no totalmente abiertas. Este es un problema que iré evolucionando y creciendo y los algoritmos serán capaces de hacer muchas más cosas a la vez, pero actualmente estos algoritmos se diseñan para resolver tareas concretas en entornos cerrados con conjuntos de datos concretos.

Por último, vamos a explorar la potencia de esta suite de algoritmos puesto que no sólo es capaz de reconocer objetos. Vamos a probar varios de estos modelos para ver la capacidad de detección de estos algoritmos. Para finalizar utilizaremos estos modelos en lugar de en imágenes, usando una webcam propia.

Para poder hacer esto, existe dentro del proyecto que hemos descargado una carpeta que se llama demo que contiene un script de Python que nos permite cargar fácilmente la configuración deseada de la red y el modelo asociado a la misma. Estos modelos veremos cómo obtenerlos a continuación.

Vamos a comenzar utilizando este script con la imagen que hemos usado previamente, pero en este caso vamos a utilizar la configuración de la red conocida como Mask-RCNN que permite no sólo detectar los objetos sino además realizar la segmentación de los mismos. Para ello necesitamos obtener el modelo, es decir descargarlo de la web oficial que es la siguiente:https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md

Este modelo es el siguiente y podemos descargarlo pulsando sobre el enlace “model”. En la siguiente imagen se recuadra cuál de todos los disponibles es el modelo y dónde se debe clicar para descargarlo. Este modelo lo guardamos en nuestra carpeta principal del proyecto dónde guardamos la imagen previa.

Hecho esto, ya tenemos todo lo necesario para usar nuestro modelo sobre la imagen o incluso con una cámara conectada al ordenador o la propia webcam del mismo.

Utilizamos el siguiente comando en nuestro terminal:

```
python demo/demo.py --config-file configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml --input perros_personas.jpg --opts MODEL.WEIGHTS model_final_f10217.pkl MODEL.DEVICE cpu
```

En el comando tenemos el script de Python que utilizaremos, el archivo de configuración en el que decimos que red neuronal y qué modo de funcionamiento deseamos. Introducimos la entrada, en este caso la ruta y nombre hacia nuestra imagen, sobre la que aplicaremos el algoritmo. Finalmente añadimos una serie de opciones adicionales como la ruta y nombre del modelo que acabamos de descargar y una opción para decir que usaremos la CPU del ordenador. Hecho esto obtenemos el siguiente resultado

Vemos ahora que no sólo se han detectado los objetos, sino que además se ha detectado la forma de los mismos pintando sobre la imagen el resultado.

Vamos a probar ahora otro modelo diferente para segmentación, pero que, en lugar de segmentar objetos aislados, este es capaz de segmentar la imagen completa y dividirla en partes. Para ello, descargamos el siguiente modelo de la misma manera que antes.

Una vez descargado ejecutamos el siguiente comando:

```
python demo/demo.py --config-file configs/COCO-PanopticSegmentation/panoptic_fpn_R_50_3x.yaml --input perros_personas.jpg --opts MODEL.WEIGHTS model_final_c10459.pkl MODEL.DEVICE cpu
```

Como se observa ahora la segmentación es diferente dónde se han diferenciado todas las partes de la imagen como la vaya, los árboles y el pavimento.

Vamos a utilizar un último modelo capaz de detectar puntos de interés del esqueleto de las personas. Descargamos entonces el siguiente modelo.

Empleamos después el siguiente comando:

```
python demo/demo.py --config-file configs/COCO-Keypoints/keypoint_rcnn_R_101_FPN_3x.yaml --input perros_personas.jpg --opts MODEL.WEIGHTS model_final_997cc7.pkl MODEL.DEVICE cpu
```

Para finalizar, vamos a emplear una webcam conectada al ordenador con uno de estos modelos para detectar directamente los objetos desde nuestra cámara en directo. Para ello reemplazamos la entrada del algoritmo y en lugar de pasarle la imagen le pasamos directamente la opción de webcam mediante el siguiente comando:

```
python demo/demo.py --config-file configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml --webcam --opts MODEL.WEIGHTS model_final_f10217.pkl MODEL.DEVICE cpu
```

De esta manera puedes emplear tu propia cámara para detectar cualquiera de las cosas que estos modelos ofrecen, segmentar imágenes o detectar los esqueletos de las personas. Esto es sólo el comienzo pues a partir de estos resultados se pueden crear numerosas aplicaciones de interés no sólo a nivel de investigación sino a nivel comercial y de producto.

DISCUSIÓN:

Para terminar este trabajo de investigación se analizarán las diferentes cuestiones desarrolladas.

Los objetivos que se planteaban al inicio han sido resueltos y desarrollados con propiedad, respondiendo a las preguntas esenciales de ¿qué es la inteligencia artificial?, ¿para qué sirve? y ¿en qué puede ayudarnos?

Para ello se han explicado conceptos básicos como definiciones básicas o la diferencia entre Machine Learning y Deep Learning, las dos ramas principales de la IA.

A continuación, en el proyecto, se escribe sobre algoritmos básicos y las redes neuronales (que se encuentran en el Deep Learning) desarrollando sus estructuras y arquitecturas a la vez que se muestran ejemplos variados y problemas concretos en los que diversas ramas de la informática se ven influenciadas.

De forma final, se incluye un tutorial de instalación de red neuronal, de forma sencilla, y en el sistema operativo más extendido en computadores a nivel mundial, Windows para demostrar que pese a las dificultades que pueden existir en los complejos algoritmos actuales, su instalación en algunos casos puede ser sencilla, de libre acceso, gratuita y útil para el usuario.

La síntesis del trabajo y sus tesis principales, responden de forma rotunda, sin necesidad de ahondar en términos complejos, a la hipótesis principal: *La Inteligencia Artificial es una de las principales ramas de la informática actual, y aumentará su influencia en el futuro debido a la gran versatilidad que presenta sobre la resolución de problemas.*

A lo largo del trabajo hemos podido corroborar estos datos, y debido a que la presencia actual de los sistemas inteligentes es inmensa y se espera que siga aumentando de manera excepcional, se puede afirmar que la inteligencia artificial es una revolución continua que mejorará nuestro nivel de vida ayudándonos a realizar diversas tareas complejas o que para un humano significarían un arduo trabajo.

BIBLIOGRAFÍA:

2017-report.pdf. (s. f.). Recuperado 22 de mayo de 2021, de http://aiindex.org/2017-report.pdf

2021-AI-Index-Report-_Chapter-3.pdf. (s. f.). Recuperado 22 de mayo de 2021, de https://aiindex.stanford.edu/wp-content/uploads/2021/03/2021-AI-Index-Report-_Chapter-3.pdf

060031.pdf. (s. f.). Recuperado 21 de mayo de 2021, de https://www.acta.es/medios/articulos/ciencias_y_tecnologia/060031.pdf

Almeida, L. B. (1990). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. *Undefined*. https://www.semanticscholar.org/paper/A-learning-rule-for-asynchronous-perceptrons-with-a-Almeida/8be3f21ab796bd9811382b560507c1c679fae37f

Altman, R. B. (1999). AI in Medicine: The Spectrum of Challenges from Managed Care to Molecular Medicine. *AI Magazine*, *20*(3), 67-67. https://doi.org/10.1609/aimag.v20i3.1467

Brain, I. T., & Rosenblatt, F. (s. f.). *The Perceptron: A Probabilistic Model for Information Storage and Organization*.

Bregman, D. (s. f.). *Smart Home Intelligence- The eHome that Learns*.

COCO - Common Objects in Context. (s. f.). Recuperado 13 de octubre de 2021, de https://cocodataset.org/#home%C3%A7

Estlin, T., Gaines, D., Chouinard, C., Castano, R., Bornstein, B., Judd, M., Nesnas, I., & Anderson, R. (2007). Increased Mars Rover Autonomy using AI Planning, Scheduling and Execution. *Proceedings 2007 IEEE International Conference on*

Robotics and Automation, 4911-4918. https://doi.org/10.1109/ROBOT.2007.364236

Gold, A. (2005). *Academic AI and Video Games: A Case Study of Incorporating Innovative Academic Research into a Video Game Prototype*. http://nn.cs.utexas.edu/?gold:cig05

Gunn, E. A. A., Craenen, B. G. W., & Hart, E. (s. f.). *A Taxonomy of Video Games and AI*.

Hassabis, D. (2017). Artificial Intelligence: Chess match of the century. *Nature*, *544*(7651), 413-414. https://doi.org/10.1038/544413a

http://www.whitehouse.gov, W. H. (2018, mayo 10). Summary of the 2018 White House Summit on Artificial Intelligence for American Industry. *Homeland Security Digital Library*. United States. Office of Science and Technology Policy. https://www.hsdl.org/?abstract&did=

Johnson, C. G. (2021). Solving the Rubik's cube with stepwise deep learning. *Expert Systems*, *38*(3), e12665. https://doi.org/10.1111/exsy.12665

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., & Houlsby, N. (2020). Big Transfer (BiT): General Visual Representation Learning. *arXiv:1912.11370 [cs]*. http://arxiv.org/abs/1912.11370

Kröse, B., Krose, B., Smagt, P. van der, & Smagt, P. (1993). *An introduction to Neural Networks*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. https://doi.org/10.1038/nature14539

McGregor, A., Hall, M., Lorier, P., & Brunskill, J. (2004). Flow Clustering Using Machine Learning Techniques. En C. Barakat & I. Pratt (Eds.), *Passive and Active Network Measurement* (pp. 205-214). Springer. https://doi.org/10.1007/978-3-540-24668-8_21

Pearlmutter, B. (1991). *Dynamic Recurrent Neural Networks*.

Piccinini, G. (2004). The First Computational Theory of Mind and Brain: A Close Look at Mcculloch and Pitts's "Logical Calculus of Ideas Immanent in Nervous Activity". *Synthese*, *141*. https://doi.org/10.1023/B:SYNT.0000043018.52445.3e

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, *59*(19), 2229-2232. https://doi.org/10.1103/PhysRevLett.59.2229

Roopa, H., & Asha, T. (2019). A Linear Model Based on Principal Component Analysis for Disease Prediction. *IEEE Access*, *7*, 105314-105318. https://doi.org/10.1109/ACCESS.2019.2931956

Seber, G. A. F., & Lee, A. J. (2003). *Linear Regression Analysis* (1.ª ed.). Wiley. https://doi.org/10.1002/9780471722199

Sémery, O. (2021). *Deep learning networks* [Python]. https://github.com/osmr/imgclsmob (Original work published 2018)

Wu, R., Yan, S., Shan, Y., Dang, Q., & Sun, G. (2015). Deep Image: Scaling up Image Recognition. *arXiv:1501.02876 [cs]*. http://arxiv.org/abs/1501.02876

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., & Wang, C. (2018). Machine Learning and Deep Learning Methods for Cybersecurity. *IEEE Access*, *6*, 35365-35381. https://doi.org/10.1109/ACCESS.2018.2836950