**UNIVERSIDAD ALFONSO X EL SABIO**

**FACULTAD DE CIENCIAS DE LA EDUCACIÓN**

**MUIA Master Universitario en Inteligencia Artificial**



**TRABAJO DE FIN DE MÁSTER**

**Redes Neuronales Automodelables**

**Carlos Bilbao Lara**

**Alberto Partida Rodríguez**

**ABRIL, 2024**

**TRABAJO FIN DE MÁSTER**

**OMPE**

**23-24**

**UAX**

RESUMEN

[Resumen de 300 palabras en castellano, siguiendo el formato IMRyD. Espera a terminar el proyecto para elaborarlo. Desde la introducción a las conclusiones describe el proyecto que has realizado. Apartado por apartado, sin nombrarlos.]

Palabras clave

[5 palabras claves para que encuentren tu trabajo cuando forme parte del repositorio de la UAX. Sirven para clasificar tu trabajo en categorías generales.]

ABSTRACT

[Resumen en inglés]

Keywords

[Palabras clave en inglés.]

\* El número mínimo de palabras del TFM es de 50000 y de 60000 como máximo. Se debe seguir el formato indicado en esta plantilla y no modificarlo.

ÍNDICE

[Capítulo 1: Introducción al Trabajo Fin de Máster 4](#_Toc166523828)

[1. Justificación del proyecto realizado 4](#_Toc166523829)

[2. Presentación del proyecto y sus contenidos 6](#_Toc166523830)

[3. Ejemplo Práctico de Uso 9](#_Toc166523831)

[Capítulo 2: Objetivos del TFM 14](#_Toc166523832)

[1. Objetivo general 14](#_Toc166523833)

[2. Objetivos específicos 14](#_Toc166523834)

[Capítulo 3: Marco Teórico 16](#_Toc166523835)

[1. Algoritmo Genéticos, ¿qué son? 17](#_Toc166523836)

[1.1. Algoritmo Genéticos: Teoría de la genética. 18](#_Toc166523837)

[1.2. Algoritmo Genéticos: Ejemplos y aplicaciones. 21](#_Toc166523838)

[1.3. Algoritmo Genéticos: Selección. 22](#_Toc166523839)

[1.4. Algoritmo Genéticos: Cruce. 23](#_Toc166523840)

[1.5. Algoritmo Genéticos: Mutación. 23](#_Toc166523841)

[2. Redes Neuronales: Historia y bases 24](#_Toc166523842)

[2.1. Redes Neuronales: CNNs 26](#_Toc166523843)

[2.2. Redes Neuronales: Funciones de Coste 28](#_Toc166523844)

[2.3. Redes Neuronales: Funciones de Activación 30](#_Toc166523845)

[2.4. Redes Neuronales: Pooling 31](#_Toc166523846)

[2.5. Redes Neuronales: Padding 31](#_Toc166523847)

[3. CNNs: casos de usos 31](#_Toc166523848)

[3.1. CNNs: Arquitecturas más conocidas 31](#_Toc166523849)

[4. Redes Auto modelables: State of the art 31](#_Toc166523850)

[4.1. Algoritmos Genéticos y Redes Neuronales 33](#_Toc166523851)

[4.2. Ejemplos de Redes Auto modelables: AutoKeras 33](#_Toc166523852)

[4.3. Ejemplos de Redes Auto modelables: Neural Network Intelligence 33](#_Toc166523853)

[Capítulo 4: Marco Metodológico 34](#_Toc166523854)

[1. Población y muestra 34](#_Toc166523855)

[2. Objetivos de investigación 34](#_Toc166523856)

[3. Variables intervinientes e instrumentos de evaluación 35](#_Toc166523857)

[4. Diseño experimental 35](#_Toc166523858)

[5. Modelo de análisis de datos 35](#_Toc166523859)

[Capítulo 5: Presentación de análisis, resultados y discusión 36](#_Toc166523860)

[1. Análisis descriptivo de la muestra 36](#_Toc166523861)

[2. Resultados obtenidos 36](#_Toc166523862)

[3. Discusión y análisis de los resultados 36](#_Toc166523863)

[Capítulo 6: Conclusiones 37](#_Toc166523864)

[1. Limitaciones 37](#_Toc166523865)

[2. Prospectiva 37](#_Toc166523866)

[3. Consideraciones finales 37](#_Toc166523867)

[Bibliografía 39](#_Toc166523868)

# Capítulo 1: Introducción al Trabajo Fin de Máster

A lo largo de este proyecto se tratará y analizará la viabilidad, así como la utilidad de las Redes Neuronales Auto modelables, en específico, las redes neuronales convolucionales, para análisis y clasificación de imágenes. Aunque si bien está centrado en este tipo de redes, puede ser extrapolado y fácilmente adaptado a cualquier otro tipo de redes.

1. Justificación del proyecto realizado

[Primero justificaremos de manera personal por qué elegimos este proyecto (¿Cómo se te ocurrió la realización de este proyecto? ¿Por qué crees que lo vas a realizar bien?), y luego justificaremos la utilidad científica/educativa del proyecto (¿Por qué crees que será útil? ¿Por qué crees que es relevante realizarlo como TFM?).

Este proyecto surgió como idea de evitar tener que hacer la tediosa fase de búsqueda de hiperparámetros y de la mejor arquitectura para cada dataset y cada problema que al que un experto en estas tecnologías tiene que enfrentarse cada día. Es por ello por lo que se buscó la manera de intentar automatizar estas tareas. A partir de aquí surgen muchas posibilidades para enfrentarse a esta automatización, como por ejemplo tener una pila de arquitecturas predefinidas, ejecutarlas y escoger la que mejor resultado te dé, hacer búsqueda de hiperparámetros con metodologías como Grid Search y con ellos ejecutas otra pila de arquitecturas predefinidas de nuevo, o por ejemplo la que se centrará este proyecto, arquitecturas que se van adaptando y modificando al dataset buscando la mejor arquitectura posible usando algoritmos genéticos.

Es realmente interesante pensar en obtener un algoritmo lo más pulido posible ya que ayudará a la comunidad a poder buscar el mejor resultado posible en los problemas a los que se enfrentarán, aunque sí que es verdad que estas ejecuciones llevarán un tiempo, en función del hardware utilizado ya que para un mismo dataset se entrenarán y evaluarán muchísimas arquitecturas.

Es por ello por lo que la profundidad del algoritmo y, por tanto, la cantidad de descendencia que se vuelve necesario permitir al usuario que pueda regular la cantidad que desea, con el fin de esperar más o menos tiempo a sus resultados, ya que para datasets más simples puedes encontrar la arquitectura óptima o lo suficientemente óptima para el usuario en unas pocas iteraciones, y sin embargo para datasets más complejos puede ser muy necesario utilizar un número más amplio de iteraciones del algoritmo para obtener un resultado suficientemente óptimo para él.

También, se ha de poder permitir al usuario cuantas epochs (número de veces que se entrenará y evaluará el algoritmo) ya que pasa exactamente lo mismo que con el número de descendencia, para datasets simples, puede necesitar solo unas pocas epochs para converger en una alta precisión en la fase de evaluación, sin embargo; para datasets más complejos harán falta una cantidad más elevada de epochs.

Con el fin de evitar un mayor tiempo de entrenamiento y el sobreajuste de determinadas arquitecturas, se introducirá un sistema de parada para que si e detecta que el modelo no está mejorando o se está sobre ajustando, se deje de entrenar y pase a la siguiente arquitectura.

Hoy en día ya existen ciertos modelos que buscan automatizar este proceso [1] como AutoKeras, este algoritmo buscará en su base arquitecturas predefinidas, bajo la API de AutoKeras una serie de arquitecturas que puedan ser las que mejor se adapten a priori al dataset. Tras ello, escogerá la mejor y la mutará para crear la siguiente configuración a evaluar. El diseño del algoritmo se basa en Hill-Climbing. En el capítulo 3 se explicará más en detalle este y otros algoritmos similares de automatización de la búsqueda de arquitecturas explicando y desarrollando el marco teórico del proyecto.

Posteriormente se explicará en el capítulo 4 que tecnologías y que metodologías se han empleado para desarrollar y completar el proyecto, así como una explicación sobre cómo se realizará la evaluación de los resultados de este. Estos se presentarán y analizarán en el capítulo 5, seguido de los errores cometidos, y una prospectiva del proyecto en el último capítulo.

Seguramente necesites contextualizar el tema de estudio en la actualidad y en el contexto donde se va a aplicar de forma muy general. Es decir, habla de lo que dicen los ámbitos científico, social y educativo hoy en día respecto al tema tratado sin mucha profundidad (CONSEJO GENERAL: Dedícale un pequeño párrafo a describir cada capítulo puede ser efectivo).]

1. Presentación del proyecto y sus contenidos

[Describe de forma simple el problema que presentas y cómo tu proyecto contribuye a resolverlo. Posteriormente, haz un recorrido por la distribución de los contenidos en tu proyecto describiendo de forma simple los capítulos y apartados que vas a incluir, de forma lógica, para preparar al lector a su lectura.]

Actualmente uno de los pasos más tediosos y largos en el desarrollo de un modelo de redes neuronales es generar unos hiperparámetros y una arquitectura que se adapten perfectamente al dataset, es decir; que permitan un entrenamiento relativamente rápido obteniendo una alta precisión en la fase de evaluación, sin sobre ajustarse. Habitualmente este paso es relativamente lento ya que, aunque se puede intuir, no se sabe exactamente como influirá al resultado del entrenamiento y testeo modificar esta u esta otra capa, añadir más o menos neuronas, añadir o eliminar capas ….

Es por ello por lo que este proyecto pretende ayudar en esta fase, automatizando todo este proceso de búsqueda, permitiendo realizar esa búsqueda manual de la mejor arquitectura e hiperparámetros a partir de Grid o Random Search para los hiperparámetros y algoritmos genéticos para la arquitectura.

Esta solución pretende obtener que ajustes son los que mejor se están adaptando al dataset del problema para obtener la mejor solución.

Aunque si bien es cierto que se automatiza esta tarea y se va a reducir el tiempo necesario para encontrar la arquitectura ideal, el tiempo necesario para encontrar la arquitectura va a seguir siendo muy elevado debido a que entrenar y evaluar múltiples arquitecturas es elevado (siempre dependiendo que hardware y que dataset se esté utilizando), es por ello que se deja al usuario indicar cuanta profundidad quiere en la búsqueda, es decir, en cuantos descendientes quiere buscar la mejor arquitectura. Para datasets muy complejos puede ser conveniente realizar una búsqueda más profunda, pero ello también llevará más tiempo.

Con el fin de entender, explicar y demostrar a solución planteada en este apartado los siguientes capítulos explicarán los siguientes puntos

1. **Capítulo 2: Objetivos del TFM**, en este apartado se desarrollará los objetivos del TFM, incluyendo los objetivos generales como el desarrollo de una red auto modelable y específicos como el desarrollo de una aplicación web que dé al usuario la mejor arquitectura, y le permita realizar algunas configuraciones de la red, como la cantidad de descendientes con los que desea probar.
2. **Capítulo 3: Marco Teórico**, en este apartado se desarrollará que son las redes neuronales, las redes neuronales convolucionales, los algoritmos genéticos, Grid y Random Search, así como funcionan su funcionamiento y uso. También se desarrollará que son las redes auto modelables y el estado del arte de esta tecnología.
3. **Capítulo 4: Marco Metodológico**, en este apartado se incluye las fuentes utilizadas, la metodología de trabajo utilizada, los datasets utilizados para la observación del funcionamiento de la red auto modelable, que hardware se ha utilizado, que herramientas se han utilizado para desarrollar el proyecto y modelo de análisis de datos para analizar los resultados obtenidos.
4. **Capítulo 5: Presentación de análisis, resultados y discusión**, en este apartado se presentarán los resultados obtenidos del modelo, así como una discusión acerca de estos y posibles mejoras del modelo. También se abordará el tema de los posibles usos del proyecto y adaptaciones para nuevos proyectos.
5. **Capítulo 6: Conclusiones**, finalmente se concluye con las limitaciones del proyecto, errores cometidos, posible prospectiva, es decir; como continuar con el proyecto y unas consideraciones finales incluyendo las reflexiones finales y personales acerca del mismo, autoevaluación y agradecimientos.
6. **Bibliografía**, contiene los enlaces y referencias usadas a lo largo de esta memoria con los diferentes, libros, papers y páginas utilizadas y leídas para comprender el contenido y desarrollar el proyecto correctamente.
7. Ejemplo Práctico de Uso

Este proyecto está destinado a aquellas personas que desean crear una red neuronal convolucional con el fin de obtener la mejor puntuación posible en la evaluación de su Red Neuronal. Sin embargo, aquí se pueden dar dos situaciones, que el usuario no tenga amplios conocimientos en cómo crear, modificar y ajustar arquitecturas de redes neuronales convolucionales; o qué no quiera o tenga mucho tiempo para proba manualmente ajustes y estar monitorizando la evolución de entrenamiento y evaluación de la red neuronal que ha construido. Por tanto, aquí el usuario se plantea, ¿no existirá una forma de obtener la mejor arquitectura posible fácilmente y sin tener que intervenir constantemente en su construcción?

En este punto es donde entra en juego este proyecto, para crear un ejemplo, supondremos que el usuario quiere crear una red neuronal convolucional que detecte que hay en la imagen. Para ello, busca un dataset como CIFAR10 y se dispone a encontrar la mejor arquitectura. Sin embargo, no dispone de los conocimientos o del tiempo para estar monitorizando y ajustando la arquitectura. Por ello, decide usar este proyecto. Simplemente indicando un usuario, un email, cuantos epochs para entrenar las arquitecturas y la descendencia, el algoritmo generado en este proyecto se encargará de encontrar la mejor arquitectura posible y entregarle un resumen de esta para que pueda replicarla y usarla. Además, el usuario simplemente debe incluir el dataset con las imágenes, ya sea que estén dividido en train and test o simplemente en las carpetas con cada clase, es decir; la estructura de carpetas puede ser:

* path + cifar10\train\truck
* path + cifar10\truck

Para ambos casos la aplicación funcionará correctamente.

Captura de pantalla de un celular

Descripción generada automáticamente

*Figura 1.3.1 Captura de pantalla de la aplicación web donde el usuario introduce la información deseada*

¿Descendencia? Sí, con el objetivo de encontrar la mejor arquitectura posible, se utiliza un algoritmo genético cuyas bases teóricas se detallarán en el Capítulo 3. Un algoritmo genético en software trata de imitar la teoría de la evolución desarrollada por Darwin en 1859. La teoría de la evolución describe un mecanismo natural para la evolución de las especies a lo largo de la historia, la selección natural. (Las bases de esta teoría que serán explicadas más en detalle en el capítulo 3) Las principales fases son la herencia, la variación genética y la selección natural.

Teniendo esto en cuenta y que el usuario ha introducido 10 generaciones y 20 epochs, el proceso que realizará el algoritmo será el siguiente:

1. Genera un dataset de entrenamiento y otro de testeo con el dataset introducido por el usuario.
2. Genera una población inicial de 10 arquitecturas.
3. Entrena 20 epochs (número que ha sido introducido por el usuario) cada arquitectura y las evalúa.
4. Las 4 arquitecturas más fuertes (con mejor puntuación en evaluación) se “reproducen” generando 4 nuevos individuos cuya genética (cantidad de capas, numero de filtros…) se basa en la de sus antecesores.
5. Se introduce aleatoriamente cambios en su genética (mutación genética).
6. Se crea una nueva población reemplazando las 4 peores arquitecturas por las nuevas que acaban de ser creadas y mantenemos las demás (selección natural).
7. Se repite desde el paso 3 otras 9 veces más para obtener un total de 10 generaciones (cantidad indicada anteriormente por el usuario).
8. Se entrega al usuario la arquitectura final con mejor puntuación de todas las generaciones.

Para poder hacer todo esto, se refleja las características de una red neuronal convolucional como “genética” es decir; un gen indica la cantidad de capas convolucionales, otro la cantidad de poolings, otro el número que se usará como tamaño del kernel, otro para las capas dropout…. El conjunto de estos genes conforma una arquitectura completa. Con el fin de encontrar la mejor arquitectura, toda la población se entrena con la misma cantidad de epochs y se evalúa, posteriormente las mejores genéticas se cruzarán, generando un nuevo individuo con la genética de sus antecesores y se le realizara alguna mutación.

Tras alcanzar el máximo de descendencia indicado por el usuario, se le entregará la arquitectura con el mejor resultado obtenido durante el algoritmo genético. Para ello, usa el botón de “Check Results”, y el servidor buscará si la respuesta ya está disponible o si todavía está procesando la información.   
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 1.3.2 Captura de pantalla que muestra la notificación cuando aún se está procesando la información*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Figura 1.3.2 Captura de pantalla que muestra la arquitectura final más óptima que ha encontrado el algoritmo (con el fin de acelerar el proceso, se ha usado 2 generaciones y 5 epochs para obtener este resultado)*

[Se recomienda un tamaño de 5 a 10 páginas para este capítulo].

# Capítulo 2: Objetivos del TFM

[En un párrafo explica brevemente los contenidos de este capítulo. Se recomienda un tamaño de 1 a 3 páginas para este capítulo.]

A lo largo de este capítulo se describen los objetivos, tanto generales como específicos propuestos para completar este proyecto.

1. Objetivo general

* [El objetivo general es lo que pretendes hacer con el TFG. El último paso a realizar. Redáctalo en infinitivo y solo un verbo.]
* El objetivo general del proyecto es crear una aplicación web que permita a cualquier usuario obtener a través del backend la mejor arquitectura encontrada para el dataset del usuario.

1. Objetivos específicos
2. Investigar otras posibles soluciones en el mercado.
3. Investigar y definir que son los algoritmos evolutivos y redes neuronales convolucionales.
4. Desarrollar una arquitectura de redes neuronales, que permita generar un modelo en base a un vector.
5. Desarrollar un algoritmo evolutivo que permita ir cruzando y mutando las diferentes arquitecturas.
6. Desarrollar una interfaz en REACT.
7. Desarrollar una arquitectura REST-API para combinar el backend con el frontend.
8. [Los objetivos no son deseos para el futuro. Cuando terminas el proyecto se deben haber cumplido.]
9. [Lo mejor es poner un objetivo para el marco teórico, otro para el metodológico, para el empírico, etc. Es decir, trocear el objetivo final en pequeñas metas volantes ;)]

# Capítulo 3: Marco Teórico

[En un párrafo explica brevemente los contenidos de este capítulo antes de la distribución en distintos apartados.

--------------------------

A lo largo de este capítulo se explicará la base teórica de este proyecto, es decir; ¿Qué son y cómo funcionan los algoritmos genéticos? ¿Qué son y cómo funcionan las redes neuronales? ¿Qué son y cómo funcionan las CNNs? Y por último una explicación de cómo funcionan las aplicaciones basadas en microservicios como la que se ha desarrollado en el proyecto. Además, un último apartado tratando el estado del arte de estas redes auto modificables.

---------------------------

En un marco teórico se debe describir la información científica relevante y para tu proyecto. Procura reflejar información actualizada y citarla adecuadamente (Normas APA 6ª edición que tienes en el Campus Virtual).

Incluye, al menos, tantos apartados como áreas en las que esté implicadas tu proyecto.

Fuentes habituales donde obtener trabajos de referencia:

* *Google académico*: <https://scholar.google.es/> como principal herramienta de búsqueda, que enlaza todas las bases de datos.
* *ERIC*: https://eric.ed.gov/
* *Psycnet*: http://psycnet.apa.org/
* *Dialnet*: https://dialnet.unirioja.es/
* *Medline/Pubmed*: https://www.nlm.nih.gov/bsd/pmresources.html
* Otros (cada ámbito científico suele tener sus propias bases de datos a consultar).

Se recomienda un tamaño de 10 a 20 páginas para este capítulo]

1. Algoritmo Genéticos, ¿qué son?

[2] Un algoritmo genético es un método de búsqueda que trata de imitar la teoría de la evolución biológica creada por Darwin para resolver problemas programáticos. Darwin describió un mecanismo natural para la evolución de las especies a lo largo de la historia conocido como selección natural. Aunque si bien es cierto que comprendió que los seres vivos tienen ciertas variaciones en sus rasgos con respecto a sus antecesores, no fue hasta Gregor Mendel que se desarrolló la teoría de la genética [3], aunque primeramente la desarrolló a finales del siglo XIX, publicando sus descubrimientos en 1864, las leyes de la herencia genética publicadas en 1866 y finalmente en 1874 su última obra, la “Teoría mecánico-fisiológico de la evolución orgánica”, mismo año en el que falleció. Sin embargo, no fue hasta el siglo XX que finalmente los biólogos comprendieron que esta teoría explicaba correctamente aquellos conceptos de la teoría de la evolución que Darwin había dejado sin resolver: cómo se produce la herencia, por qué existen variaciones entre individuos en una misma población y la selección natural.

Los algoritmos genéticos, aplicados al contexto software, basan su premisa en que las características de un individuo están codificadas en su genética, por ejemplo, un guisante es verde, porque hay un gen que dice que expresa que es verde, una persona tiene el pelo rubio y no castaño, porque tiene un gen que expresa este rasgo. Integrando tanto la teoría de la evolución de Darwin con la teoría genética de Mendel, se identifica 3 fases clave en los algoritmos genéticos:

* **Herencia**: los descendientes comparten la mayoría de las características genéticas de sus progenitores tal y cómo descubrió Mendel, aunque pueden existir ciertas mutaciones
* **Variación**: Darwin observó que los individuos dentro de una población varían en algunas características entre ellos. Tras los descubrimientos de Mendel, pasó a conocerse como Mutación Genética, es decir; aquellas diferencias que Darwin observó en los individuos de la misma población son la consecuencia de ciertas variaciones de los genes que ha obtenido de sus antecesores.
* **Selección Natural**: los individuos con las características más ventajosas para su supervivencia son los acaban predominando mientras que las otras acaban desapareciendo. La comprensión moderna de la genética ha permitido explicar cómo estas variaciones genéticas favorecen a ciertos individuos sobre los demás.
  1. Algoritmo Genéticos: Teoría de la genética.

[4] Gregor Mendel, a través de experimentar y cruzar seres vivos, entre ellos su conocido experimento con guisantes, estableció los fundamentos de la herencia genética, conocidos hoy en día como las leyes de Mendel. Para poder entender sus leyes, se explica a continuación los experimentos que realizó Mendel para desarrollar la teoría de la genética.

En los experimentos de su huerto con los guisantes, descubrió que, si juntaba 2 guisantes de la misma raza “pura”, su descendencia sería idéntica, es decir; si cruzaba 2 guisantes verdes, su descendencia sería verde, y si juntaba 2 amarillos su descendencia sería amarilla.

Gráfico

Descripción generada automáticamente con confianza baja

*Figura 3.1.1.1 [4] Representación del cruce de 2 razas puras distintas*

Al cruzar 2 guisantes puros, de diferentes razas, por ejemplo, uno amarillo y uno verde, el resultado al contrario de lo que se pudiera imaginar fue que todos eran amarillos.

Gráfico

Descripción generada automáticamente con confianza media

*Figura 3.1.1.2 [4] Representación del cruce de la descendencia de 2 razas puras distintas*

Sin embargo, si cruzamos esta segunda generación de guisantes que han salido todos amarillos, aparecerán algunos verdes en su descendencia, aproximadamente con una proporción de 3 a 1.

El hecho de que los guisantes sean todos amarillos en su primera generación se debe a que esta característica es dominante, es decir; tienda a aparecer más frecuentemente en sus descendencias. Por el contrario, el color verde es recesivo. Con todo ello, Mendel separó los fenotipos, que son las características observables, como el color del guisante, de los genotipos, que es el conjunto de genes en el ADN de un organismo. Por ejemplo, para un mismo gen P, existe también valor contrario p, si los dos antecesores tienen el mismo tipo como genotipo P, su descendencia tendrá el gen P y mostrará el fenotipo correspondiente. Si los antecesores son P y p, el fenotipo resultante será el dominante de los 2.

Diagrama, Escala de tiempo

Descripción generada automáticamente

*Figura 3.1.1.3 [4] Representación del cruce de la descendencia de 2 razas puras distintas*

Con todo ello, Mendel creó las leyes de la genética que son:

* **Ley de la uniformidad**: al cruzar una raza pura de una especie, con otra raza pura, la descendencia de la primera generación será física y genotípicamente iguales entre sí.
* **Ley de la segregación**: cada individuo posee dos alelos para cada gen, que se separan durante la formación del individuo, ya que cada gameto solo puede llevar 1 de los dos. Estas características se “escogen” aleatoriamente, y el fenotipo es el resultado de combinar estas características.
* **Ley de transmisión independiente**: la segregación de los rasgos hereditarios se da de forma independiente de unos individuos a otros, por lo que la genética de uno no afectará al patrón de herencia del otro
  1. Algoritmo Genéticos: Ejemplos y aplicaciones.

Los algoritmos genéticos se utilizan en el software para buscar soluciones óptimas a problemas complejos mediante la simulación de procesos evolutivos. [5] Entre sus principales usos se encuentra:

* Optimización de redes y sistemas: mejorando el diseño y la operación de redes complejas como redes de telecomunicación y transporte
* Finanzas: para selección de carteras y estrategias de trading
* Gestión automatizada de equipamiento industrial
* Inteligencia Artificial: encontrando soluciones en el aprendizaje automático, sobre todo se aplica en optimización de juegos y planificación de tareas
* Ingeniería y diseño: optimizando procesos industriales como en aerodinámica y configuración de sistemas
* Medicina: ayudando en el diseño de nuevos fármacos, o en optimizar modelos de propagación de enfermedades y estrategias de salud pública.
  1. Algoritmo Genéticos: Selección.

[3] Escoger el tamaño de la población inicial puede ser determinante para que la ejecución del algoritmo sea óptima. Por ejemplo; si la población inicial no recoge suficientes diferencias genéticas, podrías no explorar todas las posibilidades, sin embargo; si es demasiado grande, el algoritmo se volverá demasiado lento. Del mismo modo, una población más grande puede ayudar a encontrar una solución óptima, pero si se utiliza una población más pequeña se encontraría una solución subóptima pero no necesaria.

[7] Tras determinar la puntuación de cada individuo se procede a seleccionar aquellos individuos que se van a reproducir para generar una descendencia. Hay una gran variedad de tipos de selección, pero entre las más destacadas encontramos:

* **Selección proporcional o de ruleta**: los individuos se seleccionan según su puntuación, a mayor puntuación, mayor probabilidad de ser elegido. Sin embargo, hay que tener cuidado con esto ya que puede dar una convergencia prematura
* **Selección por torneo**: se escoge un numero pequeño de individuos al azar y se escoge al mejor del grupo. Es simple y eficaz por lo que ha sido escogido para la implementación práctica de este proyecto.
* **Selección por rango**: los individuos se clasifican por su puntuación y según el rango en el que estén se les asigna una probabilidad de ser escogido.
* **Selección elitista**: se asegura de escoger a los mejores individuos de cada generación y que se conserven para la siguiente.
* **Selección estocástica**: similar a la selección proporcional, pero reduce la probabilidad de que sean escogidos los mejores individuos múltiples veces. Garantiza de este modo una distribución más uniforme
  1. Algoritmo Genéticos: Cruce.

Consiste en combinar la genética de 2 individuos seleccionados anteriormente seleccionados para generar un nuevo individuo. Del mismo modo que antes también existen diferentes tipos de cruce, el objetivo de este paso es encontrar la mejor descendencia posible. [3] Existen diferentes tipos de cruce como:

* **Cruce de 1 punto**: se trata de copiar exactamente los mismos genes de uno de los padres hasta un punto de la cadena y a partir de ahí copia los genes del otro.
* **Cruce de 2 puntos**: similar al anterior, pero añadiendo otro punto en la cadena donde vuelva a copiar los genes del primero de los padres.
* **Cruce uniforme**: cada gen se copia de uno de los 2 padres aleatoriamente permitiendo. Este es el que se ha escogido en la implementación de este proyecto
  1. Algoritmo Genéticos: Mutación.

Tras haber generado un nuevo individuo en la fase de cruce, falta mutarlo. La mutación se hace con el fin de encontrar nuevas posibilidades de combinaciones genéticas que pueda favorecer la “supervivencia” del individuo. Gracias a ello, se previene al algoritmo de encontrar un máximo local.

Por tanto, el proceso de mutación consiste en la introducción de modificaciones en la cadena genética del individuo de forma aleatoria, atendiendo a un umbral. Este umbral determinará cuanto y cuantas veces se producirán mutaciones en la descendencia. Escoger este umbral dependerá de que se desee más, si la exploración o la explotación, con el peligro de tardar mucho en converger, o quedarse estancado en un óptimo local respectivamente.

1. Redes Neuronales: Historia y bases

[8] Las redes neuronales artificiales son un subconjunto de machine learning y están en el núcleo de los algoritmos de Deep Learning. Su nombre se debe que la idea del funcionamiento de este software se basa en imitar cómo funcionan las redes neuronales biológicas que tienen los seres humanos en el cerebro.

Aunque el boom de esta tecnología se ha dado recientemente, sobre todo con el descubrimiento al público general de tecnologías como CHAT-GPT o Dall-E 3, el inicio de las redes neuronales se remonta a 1940, cuando Warren McCulloch y Walter Pitts introdujo el concepto de red neuronal. En 1943, propusieron un modelo matemático simple para la actividad neuronal. [9] En 1951, se desarrolló lo que se considera la primera red neuronal artificial de la historia, creada por Minsky y Dean Edmunds. Por esta época, en el año 1950 Alan Turing desarrolló el conocido Test de Turing, un concepto de juego de imitación donde un humano habla con una inteligencia artificial sin saberlo, si el humano no se da cuenta de que no está hablando con una persona de verdad, si no, con una IA, esta pasa el test.

[8] Sin embargo, no fue hasta 1957 que Frank Rosenblatt desarrolló el perceptrón simple, una simple neurona que imita las capacidades de percepción del ser humano y que es conocida ya que es la base de las siguientes investigaciones de redes neuronales. El perceptrón simple es una unidad binaria de procesamiento que recibe una serie de entradas y mediante unos pesos (la importancia relativa que existe entre las entradas y la salida) y un umbral de resultado a esas entradas, el perceptrón devuelve 1 o 0.

Siguiendo con la investigación y el desarrollo de Frank Rosenblatt, casi una década después se creó el perceptrón “multilayer”, una extensión del perceptrón simple que utilizaba múltiples neuronas formando capas en 1965, cuyos valores de los pesos de cada neurona se modificaban y actualizaban manualmente. Debido a que esto era un trabajo tedioso, la investigación sobre redes neuronales se enfocó en desarrollar un algoritmo que pudiera actualizar los pesos directamente mientras la red se entrenara. Fue en 1986 cuando David Rumelhart, Geoffry Hinton y Ronald Williams desarrollaron el algoritmo de “Backpropagation”, fundamental y usado hoy en día, este algoritmo permite a la red calcular el error obtenido en la salida y propagar hacia las capas anteriores pequeños ajustes en los pesos con el fin de obtener mejores resultados en la siguiente iteración.

Poco después en el año 1989, inspirándose en la visión de los animales, desarrollaron las redes neuronales convoluciones, o en inglés, Convolutional Neuronal Networks (CNNs). La principal innovación de estás redes fue la introducción de capas convolucionales en las redes neuronales artificiales, dotándolas de “visión” y permitiendo entrenar a las redes para reconocimiento de objetos en imágenes entre otros usos. Por ejemplo, con estas redes se puede enseñar a un dron que se encuentre volando por una ciudad a esquivar los obstáculos de la calle, como edificios, coches, señales….

En 1997 se creó otra de las arquitecturas que sirve de base de las redes más utilizadas y conocidas hoy, las Long Short Term Memory (LSTM), conocidas por albergar en su arquitectura una pequeña memoria a corto plazo que se utiliza como entrada de las capas. Hoy en día son ampliamente utilizadas para tareas de NLP (Natural Lenguage Processing), clasificación de texto, generación de texto, resúmenes…, así como para predicciones de series temporales.

Casi una década después, en 2006 se desarrollaron las Deep Belief Networks, especialmente conocidas por qué demostraron que el uso de pesos aleatorios en las redes eran un error. Fueron desarrolladas por Geoffry Hinton, incluso hizo una conferencia de Google Tech Talk en 2007 explicando como funcionaban estas redes y llamándolo como “The Next Generation of Neural Networks”. Principalmente se utilizan para tareas de aprendizaje no supervisado y preentrenamiento de modelos de Deep Learning.

Por el año 2014, se presentaron las redes GAN (Generative Adversarial Networks) cuyo modelo consistía en entrenar 2 redes neuronales artificiales a la vez, uno haciendo de Generador que usando datos basura genera muestras, y Discriminador que recibe las muestras del generador y del conjunto real, teniendo que ser capaz de distinguir entre las generadas y las reales.

Por último, como redes más conocidas hoy en día, en 2017 se presentaron las redes Transformers, en el artículo Attention is All you Need [10]. El modelo fue un hito en el mundo del NLP al demostrar un rendimiento extraordinario en una gran variedad de tareas, incluyendo traducción automática de textos, y generación de texto. Entre los modelos comerciales más conocidos y que se han ganado el afecto de los consumidores están BERT de Google o GPT de OpenAI lo que ha provocado un gran auge del sector, tanto por descubrimiento entre los consumidores como el aumento en la inversión en innovación en esta área lo que provocará un gran número de nuevas arquitecturas e incluso nuevas clases de redes neuronales en el futuro.

Dado que para este tipo de tecnologías era necesario el uso de ordenadores y servidores con una gran capacidad de procesamiento, según se podía ir aumentando la capacidad, más y mejores arquitecturas se iban descubriendo, así como la cantidad de problemas y situaciones que se pueden resolver utilizando redes neuronales. Es por ello por lo que, aunque la investigación sobre redes neuronales artificiales lleve 80 años, en el último cuarto se ha avanzado muchísimo más y se han realizado una gran cantidad de descubrimientos sobre las arquitecturas de las redes y sus posibles aplicaciones, cambiando el mundo tal y como se conocía.

* 1. Redes Neuronales: CNNs

Las Redes Neuronales Convolucionales, o CNNs (Convolutional Neuronal Networks) en inglés se desarrollaron sobre el año 1989, con la intención de dotar de una visión como la de los animales a los ordenadores. Para ello se desarrollaron varias capas ocultas especializadas en las tareas de tratamiento y obtención de información de imágenes, entre ellas las capas de convolución, sampling y pooling.

Las capas convoluciones permiten a la red neuronal detectar desde las líneas más simples en una imagen hasta las formas más complejas como rostros. Para ello se necesita una gran cantidad de imágenes para que la red aprenda las formas y características de la imagen para clasificarlas correctamente, lo que supone un desafío. Además, son redes computacionalmente costosas debido a que necesitas 1 neurona por cada característica de la imagen, es decir, suponiendo que estamos clasificando imágenes de 28x28 pixeles y a color, necesitaremos en la primera capa 2532 neuronas (28x28x3 = 2532) y esta imagen sería de bastante baja calidad por lo que costaría hasta para un humano reconocer lo que hay en ella. Sin embargo, si ese número ya parece alto, realicemos el cálculo con una imagen HD, 1920x1080 píxeles y a color; la primera capa de la red necesitará 6.220.800 neuronas, es por ello por lo que se suele realizar un tratamiento a las imágenes para recudir su dimensionalidad.

Además de ello, con el fin de obtener más datos, y a su vez, reducir el posible overfitting, el dataset se suele someter, no solo a esa reducción de calidad de imagen si no también a transformaciones como rotarla, cortarla, subir la exposición, o bajarla, voltearla vertical o/y horizontalmente incluso añadir ruido, eso sí tratando que estas transformaciones permitan a la imagen seguir siendo realista y que se puedan encontrar esas mismas imágenes transformadas en la vida real. A esta técnica se la conoce como “data augmentation”.

Cada valor de cada píxel se encuentra entre 0 y 255, por lo que para simplificar cálculos y el aprendizaje en la red se estandariza y se pone el valor entre 0 y 1 dividiendo su valor por 255. Posteriormente la capa convolucional procede crea grupos de kernel, que consisten, en grupos de pixeles cercanos sobre los que se calcula su producto escalar, permitiendo reducir la complejidad de la red.

Diagrama

Descripción generada automáticamente

*Figura 3.2.1.1 [11] Agrupación en kernel*

Tras crear los kernel, se aplica una primera capa de convolución y en su salida, una función de activación, por ejemplo, ReLu que es la más extendida hoy en día. La idea de esta función de activación es discriminar los valores que no se quieran utilizar, y estandarizar aquellos que sí pasan ese valor de corte. Posteriormente se utilizan capas de subsampling o Max-Pooling para reducir la complejidad de la arquitectura. Resumidamente, el subsampling es una práctica común en la codificación de imágenes que, realizando un muestreo de frecuencia de la crominancia, se reduce el ancho de memoria necesario para procesar una imagen sin afectar a la calidad de esta; mientras que el Max-Pooling busca también reducir la complejidad buscando reducir la dimensión de la imagen aplicando filtros, habitualmente de 2x2 o 3x3, escogiendo el máximo valor de los pixeles de cada filtro y reduciendo los 2x2 o 3x3 pixeles a 1.

* 1. Redes Neuronales: Funciones de Coste

[8] Las funciones de coste determinan el error que existe entre el valor predicho por las redes neuronales y el valor real que debería ser. Por ejemplo, supongamos que estamos analizando que objeto sale en la imagen, si mostramos un coche y la red predice camión, se calcula el error con respecto a la solución esperada, que será menor y distinto a si predice árbol. De esta manera se intenta corregir a través de backpropagation los pesos y optimizar la red buscando el error sea lo más bajo posible y se vaya reduciendo a lo largo del entrenamiento de la red. Entre las funciones más utilizadas a día de hoy son:

* **Cross Entropy Loss**: esta función calcula la pérdida usando la entropía cruzada entre la entrada y el objetivo. Se utiliza para clasificación múltiple, es este tipo de casos la red devuelve un vector de valores con las probabilidades que hay de que pertenezca lo que se quiere clasificar a esa etiqueta. La función puede devolver desde 0 hasta infinito, donde se intentará buscar que sea 0 que quiere decir que la predicción realizada es correcta.
* **Binary Cross Entropy Loss**: especialización de Cross Entropy Loss para problemas de clasificación binaria, como, por ejemplo; si una imagen es un perro o un gato.
* **MSE Loss (Mean Squared Error)**: la función mide la perdida dado utilizando el error cuadrático medio entre cada uno de los elementos predichos y el objetivo. Se suele utilizar para resolver problemas de regresión en aprendizajes automáticos supervisados
* **Hinge Embedding Loss**: para medir la pérdida la función utiliza un tensor x y un tensor y con valores de 1 a -1 y una con la predicción y la otra con el valor real. Busca que la distancia entre las 2 entredas sean iguales, utilizando una medición por pares:

**Texto

Descripción generada automáticamente**

*Figura 3.2.2.1 [8] Fórmula de la función de HE Loss*

* 1. Redes Neuronales: Funciones de Activación

[8] Las funciones de activación son aquellas que se encargan de definir cómo piensa una neurona de la red neuronal, ya que, son las que determinan cual será el valor de la salida de la neurona en función de la entrada que reciba. Generalmente el valor suele ir entre -1 y 1. Entre las más utilizadas hoy en día se encuentran:

* Función de Heaviside: también conocida como función de escalón unitario, fue la primera en ser utilizada en las redes neuronales debido a su simplicidad. Simplemente, devolverá 0 para todos los valores calculados hasta que sobrepasen el umbral por ejemplo de 0.5, si es superior a este número, devolverá 1.
* Función ReLU (Rectificied Lineal Unit): transforma los valores introducidos, eliminando y devolviendo 0 para aquellos valores que sean negativos y devolviendo el mismo valor que se introdujo a la función si este es positivo

Gráfico, Gráfico de líneas

Descripción generada automáticamente

*Figura 3.2.3.1 [8] Representación de la función de activación ReLU*

* Funciones Sigmoideas: aplican 2 asíntotas horizontales impidiendo que los valores tomen valores inferiores o superiores al rango (-1, 1). De este modo se mitigan los efectos de los outliers.

Diagrama

Descripción generada automáticamente

*Figura 3.2.3.2 [8] Fórmula de las funciones sigmoideas*

* Función logística: convierte en un valor entre (0, 1) casi cualquier entrada, especialmente útil en problemas de clasificación binaria como función de activación de la última capa.
* Función Softmax: o función exponencial normalizada, es una generalización de la función logística
  1. Redes Neuronales: Pooling
  2. Redes Neuronales: Padding

1. CNNs: casos de usos
   1. CNNs: Arquitecturas más conocidas
2. Redes Auto modelables: State of the art

Las redes auto modelables provienen de la rama del AutoML, o, Auto Machine Learning. A raíz de la necesidad de ir creando nuevas redes neuronales, para cada dataset, los científicos e ingenieros dedicados a este campo empezaron a desarrollar nuevas teorías y formas de conseguir crear las redes neuronales adaptadas al dataset con la menor intervención posible por parte del usuario, llegando así también a un público menos experto en esta materia.

Es por ello por lo que a partir del año 2010 comienzan a desarrollarse e investigarse posibles formas de automatizar el proceso de creación de modelos de machine learning, de los cuales se pueden destacar papers como [12] “Neural Architecture Search with Reinforcement Learning” del 2016, que es un punto de inflexión sobre el uso del AutoML en redes neuronales. En este paper utilizan reinforcement learning con el fin de encontrar la mejor arquitectura de redes neuronales a los datasets. Para ello utilizan el dataset CIFAR-10. Según ellos mismos, la búsqueda de la mejor arquitectura es autoregresiva lo que quiere decir que busca los mejores hiperparámetros de uno en uno, una idea tomada de la publicación de “Sutskever et al., 2014”. Sin embargo, el método que proponen ellos es diferencial en que optimiza, no cada hiperparámetro si no la precisión de la red hija, algo que se asemeja a la BLEU Optimization de “Ran- zato et al., 2015”; “Shen et al., 2016”, esta método propeno aprender y optimizarse en base a una recompensa. En resumen, proponen el uso de Reinforcement Learning para una pieza de su arquitectura llamada controlador, que es una red neuronal que propone posibles arquitecturas en base a su aprendizaje por refuerzo que son evaluadas para determinar que rendimiento tienen. En sus experimentos destacan que este algoritmo es capaz de proponer arquitecturas de redes neuronales muy rápido y con un rendimiento competitivo.

Por otro lado en 2018 se publico [13] “Efficient neural architecture search via parameters sharing” que aborda la importancia del uso de AutoML en las redes neuronales, para ello propone un nuevo método, ENAS (Efficient Neuronal Architecture Search), que trata de compartir los parámetros entre diferentes arquitecturas en la búsqueda y buscan optimizar la arquitectura a través del reinforcement learning. Como controlador utilizan un LSTM y de dataset CIFAR-10 y destacan que a diferencia de otros modelos existentes con una solución similar, su solución es más eficiente y económica.

El reinforcement learning así como variaciones que utilizan esta solución para encontrar la mejor arquitectura no son las únicas que se han investigado, por ejemplo [14] “Auto-Keras: An Efficient Neural Architecture Search System” en 2019, propone la búsqueda a través de un aprendizaje bayesiano.

* 1. Algoritmos Genéticos y Redes Neuronales
  2. Ejemplos de Redes Auto modelables: AutoKeras
  3. Ejemplos de Redes Auto modelables: Neural Network Intelligence

# Capítulo 4: Marco Metodológico

[En un párrafo explica brevemente los contenidos de este capítulo antes de la distribución en distintos apartados.

Describe el método que has seguido para desarrollar el proyecto. Este apartado debe contener las principales características de la investigación desarrollada, en cualquier tipo de TFM. Se debe describir el proceso que implica la detección de la necesidad o interés del trabajo, la búsqueda de la información y las fuentes documentales empleadas, el tipo de investigación (cualitativa, cuantitativa o sociocrítica, de tipo descriptiva, observación, analítica, estudio de casos, intervención, etnográfica, comparada, proyecto, investigación-acción, buena práctica, etc.). Se trata de etiquetar y catalogar el proceso, diseño y tipo de investigación que has desarrollado.

En este punto, explicaremos cómo se ha llevado a cabo la investigación del Trabajo Fin de Máster, los pasos que hemos seguido, los materiales, recursos que hemos diseñado o utilizado para proceder a la recogida de la información y la puesta en práctica de todo lo que se haya diseñado para tal fin.

Los apartados a considerar (aunque puedes variarlo para adecuarlo a tu proyecto concreto, ampliando o reduciendo apartados) son los siguientes:]

1. Población y muestra

[Describe la población objeto de estudio y el muestreo realizado para obtener la muestra que representa dicha población.]

1. Objetivos de investigación

[¿Qué pretendes evaluar? Si los objetivos son los mismos que los objetivos del TFM no hace falta que incluyas este apartado.]

1. Variables intervinientes e instrumentos de evaluación

[Describe qué vas a medir y a través de qué instrumentos]

1. Diseño experimental

[Describe el método de investigación elegido y su procedimiento]

1. Modelo de análisis de datos

[Describe cómo vas a analizar los datos que obtengas en tu investigación. Si los vas a analizar cuantitativa o cualitativamente. Si vas a realizar análisis estadísticos y mediante qué programas informáticos, etc.].

[Se recomienda un tamaño de 5 a 15 páginas para este capítulo]

# Capítulo 5: Presentación de análisis, resultados y discusión

[En un párrafo explica brevemente los contenidos de este capítulo antes de la distribución en distintos apartados.

Se trata de la descripción de los resultados obtenidos y del trabajo de campo realizado.

Se recomienda un tamaño de 5 a 15 páginas para este capítulo, pero depende del proyecto en sí y de los datos obtenidos, por lo que la variabilidad posible es amplia.

Los posibles apartados a incluir son los siguientes:]

1. Análisis descriptivo de la muestra

Describe la muestra seleccionada en función de las distintas variables consideradas y otros datos sociodemográficos de interés.

1. Resultados obtenidos

En este apartado se muestran los resultados obtenidos tras la investigación.

1. Discusión y análisis de los resultados

Interpretación del significado de los resultados obtenidos y análisis de su utilidad futura.

# Capítulo 6: Conclusiones

[Recupera los objetivos iniciales y explica si se han cumplido y por qué.

Debes comenzar con el primer objetivo específico y terminar con el general, explicando cómo y por qué has cumplido con el 100% de lo que te planteaste al inicio del proyecto.]

1. Limitaciones

[Errores cometidos y limitaciones de los resultados obtenidos, es decir, qué ha fallado para que tu proyecto no fuera perfecto.]

1. Prospectiva

[La prospectiva ¿Cómo vas a continuar/completar este proyecto? Aunque nunca lo hagas, debes decir cómo deberías hacerlo.]

1. Consideraciones finales

[Reflexión personal sobre las competencias adquiridas, lo que se ha aprendido a lo largo de todo el Máster y con la elaboración del TFM (en cuanto a contenidos y a nivel personal). Autoevaluación y propuestas de mejora en la percepción del rol docente.

Trata de responder a las siguientes cuestiones:

1. Reflexión final sobre los resultados obtenidos en el proyecto a todos los niveles.
2. Reflexión personal sobre las competencias adquiridas a lo largo del TFM, tanto a nivel profesional como del método científico.
3. Reflexión personal sobre lo que se ha aprendido a lo largo de todo el Máster, tanto a nivel profesional como personal.
4. Autoevaluación y propuestas de mejora en la percepción del rol docente.
5. Agradecimientos a todos los que han hecho posible el proyecto.
6. Deseo positivo final relacionado con el ámbito al que se ha dedicado el trabajo como colofón a todo el proyecto.

Ejemplo: “No puedo terminar este proyecto sin reflejar mi deseo de que la situación educativa de X mejore en el futuro, para lo cual será muy importante Y. Por mi parte espero que esta pequeña contribución que ha supuesto este TFM sea el inicio de Z…”]

Se recomienda un tamaño de 5 a 10 páginas para este capítulo.

# Bibliografía

[En formato APA, como ves en los siguientes ejemplos, y con sangría francesa (aplica el estilo **Bibliografía APA**: En la configuración de párrafo puedes seleccionarla por defecto en el apartado de Inicio de Word):]

Sánchez-Cabrero, R., Barrientos-Fernández, A., Arigita-García, A. Mañoso-Pacheco, L., Costa-Román, O. (2018). Demographic data, habits of use and personal impression of the first generation of users of virtual reality viewers in Spain. *Data in Brief*, 21, 2651-2657. <https://doi.org/10.1016/j.dib.2018.11.127>.

Sánchez-Cabrero, R., Barrientos-Fernández, A., & Arigita-García, A. (2018). La madurez ante el espejo: Interrelación entre la satisfacción corporal y vital de personas mayores en España. *European Journal Of Health Research*, 4(2), 67-77. <https://doi.org/10.30552/ejhr.v4i2.105> ]

[1] Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. "AutoKeras: An AutoML Library for Deep Learning." the Journal of machine Learning research 6 (2023): 1-6 <https://jmlr.org/papers/v24/20-1355.html>

[2] Arranz de la Peña, Jorge & Parra Truyol, Antonio (2007). Algoritmos Genéticos. <https://d1wqtxts1xzle7.cloudfront.net/35889597/Algoritmos_Geneticos_app-libre.pdf?1418170375=&response-content-disposition=inline%3B+filename%3DALGORITMOS_GENETICOS.pdf&Expires=1713384155&Signature=IRuY4748-itaLM5KmNMvp1OM3IBqw0W3JHpvvRf4wIkDwMZrJj9mD8RkGxeiwvnCGv91bia10gW-7m2IbtZpOyzsO9ljhYRQWA0xN7-KbM9DRFAjc80VbNANv47OoAUEnfuWP7je42jsaj6qo3OWIrFNoAKCCRQ1htc-upeJvpDrEm~ZGaZpqzYKzV7Jj9ZIuJgJrW5bT2XWTBA276U0qkTMfIlzq~~sLCswdtOCB4IPGCIkRytVhiipCDrQ8SbbyTVuEMm~MyKV~QyQy4aI2H9y~r6EltjJuWff9OnQ-mHylR5Wvlo8ZqYEoXCVvz297zlM2vKWnQ~zQ2OBbltnDw__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>

[3] Suárez Fernando & Ordóñez, Adriana (2010) De Gregor Mendel y la docencia sin licencia <https://revistas.javeriana.edu.co/index.php/vnimedica/article/download/16052/12846>

[4] Miko, I. (2008). Gregor Mendel and the principles of inheritance. Nature Education, 1(1), 134. <https://hagenetics.org/hh/wp-content/uploads/2015/04/371_Lec4.pdf>

[5] Roeva, O. (Ed.). (2012). Real-world applications of genetic algorithms. BoD–Books on Demand. [https://books.google.es/books?hl=es&lr=&id=L-KdDwAAQBAJ&oi=fnd&pg=PR11&dq=Real-World+Applications+of+Genetic+Algorithms&ots=jfocxvHUSp&sig=I7qVdhx4\_\_BgB4os03oadya7MmY#v=onepage&q=Real-World%20Applications%20of%20Genetic%20Algorithms&f=false](https://books.google.es/books?hl=es&lr=&id=L-KdDwAAQBAJ&oi=fnd&pg=PR11&dq=Real-World+Applications+of+Genetic+Algorithms&ots=jfocxvHUSp&sig=I7qVdhx4__BgB4os03oadya7MmY%23v=onepage&q=Real-World%20Applications%20of%20Genetic%20Algorithms&f=false)

[6] Bodenhofer, U. (2003). Genetic algorithms: theory and applications. Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter, 2004. <https://www.flll.jku.at/div/teaching/Ga/notes.pdf>

[7] Schmitt, L. M. (2001). Theory of genetic algorithms. Theoretical Computer Science, 259(1-2), 1-61. <https://www.sciencedirect.com/science/article/pii/S0304397500004060>

[8] Bilbao Lara, Carlos (2022). Modelización de una red neuronal capaz de detectar emoticonos en imágenes. Proyecto Fin de Carrera / Trabajo Fin de Grado, E.T.S. de Ingenieros Informáticos (UPM) 2-15 <https://oa.upm.es/71006/>

[9] Park, W. J., & Park, J. B. (2018). History and application of artificial neural networks in dentistry. European journal of dentistry, 12(04), 594-601. <https://www.thieme-connect.com/products/ejournals/html/10.4103/ejd.ejd_325_18>

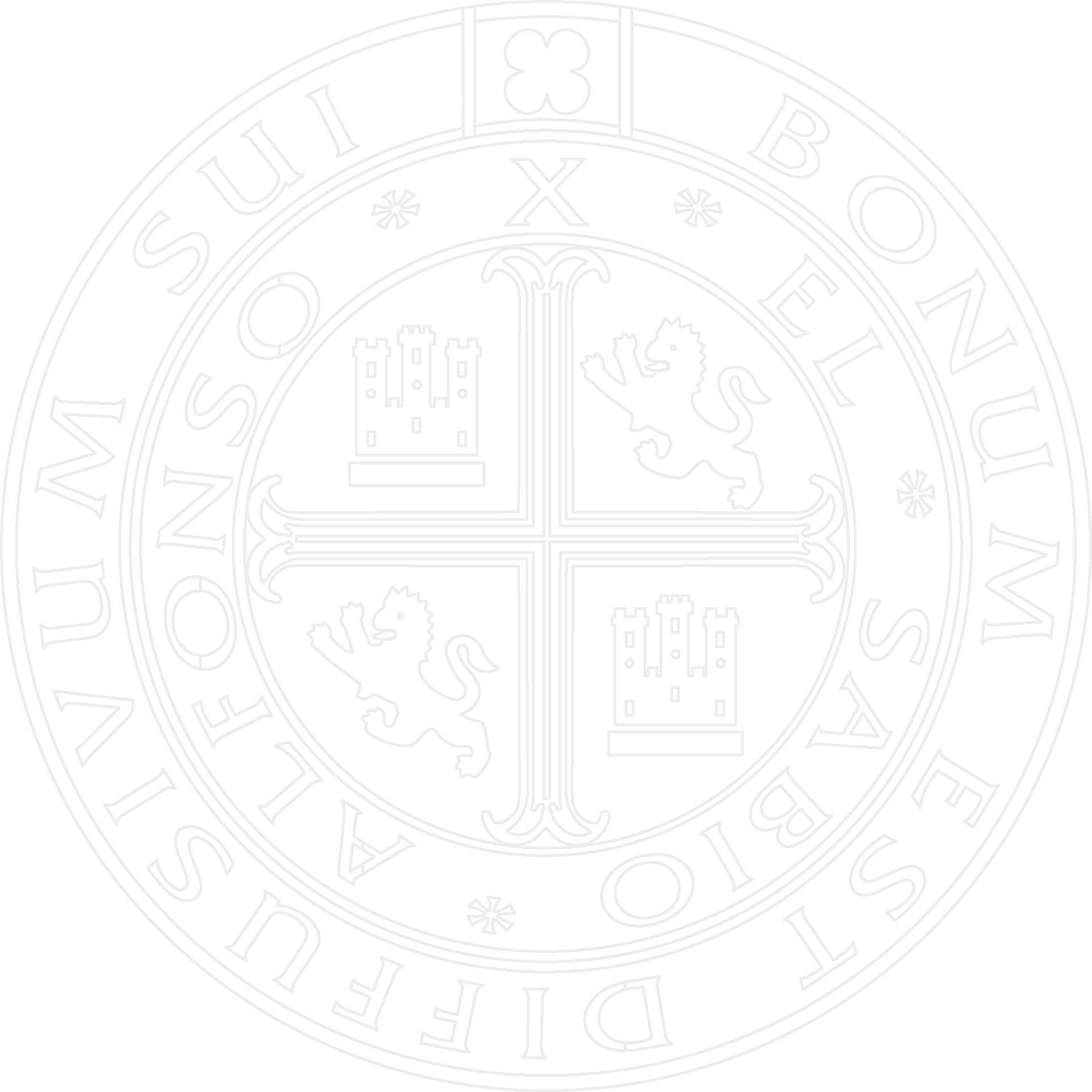
[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30. <https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>

[11] Mairal, J., Koniusz, P., Harchaoui, Z., & Schmid, C. (2014). Convolutional kernel networks. *Advances in neural information processing systems*, *27*. <https://proceedings.neurips.cc/paper_files/paper/2014/hash/81ca0262c82e712e50c580c032d99b60-Abstract.html>

[12] Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578. <https://arxiv.org/pdf/1611.01578>

[13] Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International conference on machine learning* (pp. 4095-4104). PMLR. <http://proceedings.mlr.press/v80/pham18a.html>

[14] Jin, H., Song, Q., & Hu, X. (2019, July). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1946-1956). <https://dl.acm.org/doi/abs/10.1145/3292500.3330648>



Anexos

FORMATO LIBRE

Los ANEXOS no puntúan en el trabajo, por lo que solo se recomienda incluirlos cuando son estrictamente necesarios.

Se estima una referencia máxima de un 20% del contenido del trabajo para no considerar desproporcionados los anexos. Por ejemplo: 10 páginas de anexos en un proyecto de 50 páginas totales del proyecto en sí.

Debe haber un motivo claro para incluir los anexos. Éste, en la mayoría de los casos será: “Sin este anexo no se comprende de la misma forma el trabajo, o la comprensión del trabajo se muestra sesgada o incompleta”.

Ten cuidado con el nivel de plagio, que puede verse seriamente perjudicado por lo que incluyas en los ANEXOS.

No incluyas nunca material de otros autores sin permiso expreso.

No incluyas nunca datos personales o que rompan el derecho al anonimato de los alumnos o las personas implicadas en el proyecto.

Recuerda que en los anexos es mejor una imagen que mil palabras.