UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA INFORMATICA

Algoritmos meméticos para reducir datos de entrenamiento en modelos de aprendizaje profundo convolucionales

---

Autor

José Ruiz López (alumno)

Directores

Daniel Molina Cabrera (tutor)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, Noviembre de 2024

Algoritmos meméticos para reducir datos de entrenamiento en modelos de aprendizaje profundo convolucionales

José Ruiz López (alumno)

Palabras clave: Algoritmos meméticos, Imágenes, Modelos de Aprendizaje profundo convolucionales

Resumen

Los **modelos de Aprendizaje Profundo** (Deep Learning) han supuesto un verdadero hito en la **Inteligencia Artificial**, ya que son capaces de procesar grandes volúmenes de datos... y, además, reconocer patrones sumamente complejos. Dentro de estos, los **modelos convolucionales** se han destacado como particularmente efectivos a la hora de identificar objetos y características en imágenes —una capacidad esencial para muchas aplicaciones modernas—. Sin embargo, a diferencia de los seres humanos, estos modelos requieren un número extremadamente alto de datos de entrenamiento para cada categoría que deben aprender. Esto implica un proceso de entrenamiento más largo y, muchas veces, la recolección de los datos necesarios puede ser problemática, según el tipo de información que se necesite.

Además de la dificultad en la obtención de datos, la reciente **legislación europea sobre IA** (IA Act) [] establece la necesidad de auditar no solo los modelos, sino también los datos utilizados para entrenarlos, especialmente cuando se trata de aplicaciones de IA que manejan datos sensibles. Estas auditorías, por su propia naturaleza, se volverán más complejas conforme aumente el tamaño del conjunto de entrenamiento. Por lo tanto, se vuelve completamente necesario desarrollar estrategias que permitan **reducir el tamaño de los conjuntos de datos de entrenamiento**... sin comprometer la calidad del modelo.

Ya se ha demostrado que aumentar el número de imágenes de entrenamiento puede, en ciertos casos, mejorar el proceso; sin embargo, esto solo sucede cuando las imágenes adicionales realmente contribuyen al aprendizaje. De hecho, las **técnicas de aumento de datos** (Data Augmentation) permiten reducir la necesidad de muchas imágenes similares, ya que estas pueden generarse de manera automática a partir de las ya existentes... lo que además evita problemas legales asociados a la autoría de los datos.

En este trabajo, proponemos el uso de **algoritmos meméticos** combinados con **métricas de similitud entre imágenes** para establecer un

proceso de reducción del conjunto de **entrenamiento** —lo que se conoce como **selección de instancias**—. La idea es seleccionar un conjunto reducido de imágenes representativas que, junto con las técnicas de aumento de datos, sean suficientes para entrenar modelos convolucionales con una calidad óptima. De este modo, se podría reducir significativamente el tamaño del conjunto de entrenamiento, manteniendo la calidad del aprendizaje y, a su vez, facilitando tanto el proceso de auditoría como la eficiencia computacional del sistema.

Memetic Algorithms for Reducing Training Data in Convolutional Deep Learning Models

José, Ruiz López (student)

Keywords: Memetic Algorithms, Images, Convolutional Deep Learning Models

Abstract

Deep Learning models have marked a true milestone in **Artificial Intelligence**, as they are capable of processing large volumes of data... and, moreover, recognizing highly complex patterns. Among these, **convolutional models** have proven to be particularly effective in identifying objects and characteristics in images — an essential capability for many modern applications. However, unlike humans, these models require an extremely high number of training data for each category they need to learn. This implies a longer training process, and often, the collection of the necessary data can be problematic, depending on the type of information required.

In addition to the difficulty of obtaining data, the recent **European AI legislation** (IA Act) [] establishes the need to audit not only the models but also the data used to train them, especially when dealing with AI applications that handle sensitive data. These audits, by their very nature, will become more complex as the size of the training set increases. Therefore, it becomes completely necessary to develop strategies that allow for **reducing the size of training datasets**... without compromising the quality of the model.

It has already been demonstrated that increasing the number of training images can, in certain cases, improve the process; however, this only happens when the additional images actually contribute to learning. In fact, **Data Augmentation techniques** help reduce the need for many similar images, as these can be automatically generated from existing ones... which also avoids legal issues related to data authorship.

In this work, we propose the use of **memetic algorithms**, combined with **image similarity metrics**, to establish a process of **training dataset reduction** — known as **instance selection**. The idea is to select a reduced set of representative images that, along with data augmentation techniques, are sufficient to train convolutional models with optimal quality.

In this way, the size of the training dataset could be significantly reduced, while maintaining the quality of learning and, at the same time, facilitating both the auditing process and the computational efficiency of the system.

Yo, **José Ruiz López**, alumno de la titulación INGENIERÍA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI **77964364E**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: José Ruiz López

Granada a X de mes de 201 .

D. **Daniel Molina Cabrera (tutor**, Profesor del Departamento Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Algoritmos meméticos para reducir datos de entrenamiento en modelos de aprendizaje profundo convolucionales***, ha sido realizado bajo su supervisión por **José Ruiz López (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Daniel Molina Cabrera (tutor)

Agradecimientos

Poner aquí agradecimientos...

Índice general

Capítulo 1

Introducción

1.1. Contexto

En la actualidad, nos encontramos en una era caracterizada por la generación de datos, que crece a un ritmo sin precedentes. Este fenómeno plantea la necesidad de desarrollar métodos capaces de procesar y analizar grandes volúmenes de información de manera eficiente. Los **modelos de aprendizaje profundo** —especialmente las **redes neuronales convolucionales** (CNNs)— han demostrado ser particularmente efectivos en tareas como la **clasificación de imágenes**, el **reconocimiento de patrones** y otras aplicaciones complejas. Sin embargo, el entrenamiento de estos modelos requiere numerosas cantidades de datos, lo que trae consigo importantes desafíos relacionados con el **tiempo de procesamiento**, el **costo económico** y el **consumo de recursos computacionales**.

A medida que los modelos de inteligencia artificial avanzan hacia configuraciones más complejas y precisas, la necesidad de disponer de datos de entrenamiento adecuados y en gran cantidad se hace cada vez más evidente. No obstante, la recolección, almacenamiento y procesamiento de estos datos representan barreras significativas para muchas organizaciones —especialmente aquellas con recursos limitados—, lo que subraya la importancia de explorar enfoques innovadores que permitan la **optimización y reducción** de los conjuntos de datos necesarios para entrenar estos modelos.

1.2. Motivación

La necesidad de **reducir los conjuntos de datos de entrenamiento** surge de la búsqueda por optimizar la **eficiencia** en el desarrollo de modelos de aprendizaje profundo. Si bien las redes neuronales convolucionales

han alcanzado resultados impresionantes, sus **altos costos computacionales** y la extensa cantidad de datos que requieren suponen limitaciones para muchos entornos. Entrenar estos modelos utilizando solo una parte de los datos —seleccionados de manera óptima— podría reducir considerablemente los recursos necesarios (sin comprometer la precisión de los resultados); ello marcaría un avance significativo para la **inteligencia artificial** y sus aplicaciones en entornos con **restricciones de recursos**.

Es aquí donde los **algoritmos meméticos** entran en juego: estos algoritmos combinan las fortalezas de las **técnicas evolutivas** y los **métodos de búsqueda local**, con el fin de optimizar de manera eficiente los conjuntos de datos. La selección de subconjuntos de datos más representativos permite reducir los tiempos de procesamiento y el consumo de recursos sin afectar el rendimiento de los modelos. Así, los algoritmos meméticos se posicionan como una solución prometedora para hacer que el aprendizaje profundo sea más **accesible y eficiente** —incluso en escenarios donde los recursos son limitados—, lo cual es crucial para democratizar el uso de la inteligencia artificial en una amplia gama de aplicaciones.

Este trabajo de fin de grado busca contribuir a esta dirección, explorando la aplicación de algoritmos meméticos en la reducción de datos. La investigación pretende abrir nuevas vías para el desarrollo de modelos más **eficientes**, accesibles y económicamente viables, lo que favorecerá un futuro en el que la inteligencia artificial sea **inclusiva** y más **sostenible**.

1.3. Objetivos

El objetivo principal de este TFG es investigar la aplicación de **algoritmos meméticos** para la **reducción de conjuntos de datos de entrenamiento** en modelos de **aprendizaje profundo convolucionales**. Este estudio permitirá evaluar el impacto de dichos algoritmos en la **eficiencia computacional** y en el **rendimiento de los modelos**. Para cumplir con este objetivo general, se plantean los siguientes **objetivos específicos**:

- **Desarrollar** e implementar algoritmos meméticos que seleccionen subconjuntos óptimos de datos de entrenamiento, con el fin de reducir el volumen de datos requerido para entrenar modelos convolucionales —sin comprometer la precisión de los resultados—.
- **Evaluar** el impacto de la reducción de datos en el rendimiento de los modelos, comparando aspectos clave como la precisión, eficacia y el

tiempo de entrenamiento —en modelos entrenados con conjuntos de datos completos frente a conjuntos reducidos—.

- **Optimizar la eficiencia** del entrenamiento de redes neuronales convolucionales mediante el uso de algoritmos meméticos, analizando los beneficios en términos de reducción de tiempo y costo computacional.
- **Contribuir al avance** de soluciones innovadoras en el campo del aprendizaje profundo, especialmente en escenarios con limitaciones de datos; facilitando así el acceso a esta tecnología a sectores que, de otro modo, tendrían dificultades para implementarla de manera efectiva.

A través de este estudio, se busca no solo mejorar el rendimiento y la eficiencia de los modelos convolucionales, sino también fomentar el desarrollo de soluciones más **sostenibles y accesibles** en el ámbito de la inteligencia artificial.

Capítulo 2

Metodología y Presupuesto

Aqui debo poner que he usado metodologia agil y eso.
Y realizar el presupuesto.

Capítulo 3

Fundamentos de Aprendizaje Profundo

3.1. Definición de Aprendizaje Profundo

El **aprendizaje profundo** (Deep Learning) es una subcategoría del aprendizaje automático que se basa en el uso de **redes neuronales artificiales** con muchas capas (de ahí el término "profundo"). Estas redes están diseñadas para imitar el funcionamiento del cerebro humano, lo que les permite aprender representaciones complejas de los datos de manera jerárquica.

La principal diferencia entre el **aprendizaje automático tradicional** y el aprendizaje profundo es la manera en que se manejan las características de los datos:

- En los enfoques tradicionales, el ingeniero o científico de datos debe extraer manualmente las características más importantes para entrenar al modelo (por ejemplo, bordes, formas, texturas en imágenes). En el aprendizaje profundo, las redes neuronales son capaces de **aprender automáticamente las representaciones de los datos** a partir de los datos crudos (por ejemplo, imágenes, texto, sonido).

- Este proceso es denominado **aprendizaje de características** (feature learning), lo que reduce la necesidad de intervención humana.

El aprendizaje profundo ha mostrado un rendimiento sobresaliente en diversas tareas, como el reconocimiento de imágenes, el procesamiento del lenguaje natural, la conducción autónoma y el diagnóstico médico, gracias a su capacidad para **capturar patrones complejos** en grandes volúmenes de datos.

3.2. Redes Neuronales Artificiales

Las **redes neuronales artificiales** (ANN) son el corazón del aprendizaje profundo. Estas redes están compuestas por neuronas artificiales, que son unidades matemáticas inspiradas en las neuronas biológicas. Cada neurona toma varias entradas, las procesa mediante una **función de activación**, y produce una salida. Cuando se combinan muchas de estas neuronas en capas, forman una red neuronal.

Componentes de una Red Neuronal

1. **Neuronas o Unidades**: Cada **neurona** realiza una operación simple: recibe varias entradas, las pondera por medio de **pesos** w_i, suma estos valores junto con un **sesgo** b, y aplica una función de activación. La salida de la neurona se expresa como:

   $z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b_z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$

   Luego, el valor z pasa por una función de activación, que introduce la no linealidad en el sistema, permitiendo que las redes neuronales modelen relaciones complejas.

2. **Capas de la Red**:

   - **Capa de entrada**: Es la primera capa de la red neuronal, que recibe los datos crudos (por ejemplo, píxeles de una imagen).
   - **Capas ocultas**: Estas capas intermedias entre la entrada y la salida aprenden representaciones abstractas de los datos. En una red profunda, hay múltiples capas ocultas, lo que permite la **transformación jerárquica** de los datos.
   - **Capa de salida**: Produce la predicción final, que puede ser una clase (en problemas de clasificación) o un valor numérico (en problemas de regresión).

3. **Pesos y Bias**: Los **pesos** son parámetros ajustables que determinan la importancia de cada entrada en la neurona. El **bias** es otro parámetro que se suma al valor ponderado para desplazar la activación de la neurona y permitir que el modelo ajuste mejor los datos.

4. **Funciones de Activación**: Las funciones de activación son fundamentales para que las redes neuronales puedan aprender relaciones no lineales. Entre las más comunes se encuentran:

   - **ReLU(Rectified Linear Unit)**: $ReLU(x) = \text{máx}(0, x)$, que activa solo valores positivos.
   - **Sigmoide**: Que transforma los valores en un rango entre 0 y 1.

- **Tanh (Tangente hiperbólica)**: Transforma los valores en un rango entre -1 y 1.

El uso de **backpropagation** o retropropagación permite ajustar los pesos y biases durante el entrenamiento mediante un algoritmo de optimización, como el descenso de gradiente. De esta manera, la red aprende minimizando la diferencia entre sus predicciones y las respuestas correctas.

3.3. Redes Neuronales Convolucionales

Las **Redes Neuronales Convolucionales** (Convolutional Neural Networks, CNNs) son una clase de redes neuronales profundas especialmente efectivas para el procesamiento de datos que tienen una estructura de tipo rejilla, como las imágenes. Fueron inspiradas por el sistema visual de los mamíferos, donde diferentes capas de neuronas responden a estímulos visuales de manera jerárquica.

Las CNNs son ampliamente utilizadas en tareas de **visión por computador**, como el reconocimiento de imágenes, la segmentación de objetos y la clasificación de imágenes. Lo que diferencia a las CNNs de las redes neuronales tradicionales es su capacidad para detectar **patrones espaciales** como bordes, texturas, y formas, sin necesidad de un procesamiento manual de las características.

Componentes principales de una CNN

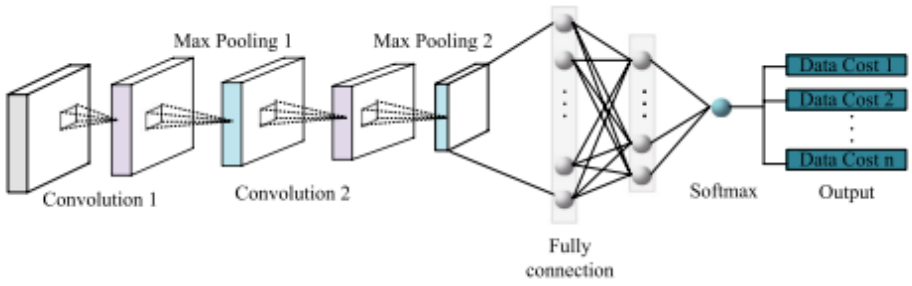

Figura 3.1: En esta imagen extraída de [**?**] puede observarse de la estructura de una red neuronale convolucional. Junto a sus capas convolucionales.

1. **Capas Convolucionales**: Estas capas aplican **filtros o kernels** sobre las imágenes de entrada para detectar características locales, como bordes, esquinas o texturas. Un filtro convolucional es una pequeña

matriz que se mueve a lo largo de la imagen, calculando productos escalares en cada posición para producir un mapa de características.

Las convoluciones son útiles porque explotan la **localidad de las características**, es decir, las relaciones espaciales entre píxeles cercanos. Además, la cantidad de parámetros se reduce drásticamente en comparación con las capas densas, ya que el filtro se comparte a lo largo de la imagen.

2. **Pooling (Submuestreo o Agrupamiento)**: Las capas de pooling reducen la dimensionalidad de las características extraídas por las capas convolucionales, lo que hace que las representaciones sean más manejables y robustas frente a pequeños cambios o desplazamientos en la imagen.

   El max-pooling es la técnica de pooling más común, donde se toma el valor máximo dentro de una ventana de píxeles, reduciendo el tamaño de la imagen, pero reteniendo las características más importantes.

3. **Capas Densas**: Después de varias capas convolucionales y de pooling, se agregan una o más **capas densas** (fully connected) para realizar la clasificación o predicción final. Estas capas toman todas las características aprendidas en las capas convolucionales y las combinan para generar una decisión final.

4. **Batch Normalization**: Esta técnica se utiliza para **normalizar** las salidas de las capas intermedias de una red neuronal. Batch Normalization ayuda a **acelerar el entrenamiento** y a hacer que la red sea más estable, al reducir el **desplazamiento covariante** (cambios en las distribuciones de las entradas de las capas intermedias a lo largo del entrenamiento). Esto se logra al normalizar las entradas de cada capa convolucional o densa antes de aplicar la activación, ajustando su media y varianza.

5. **Dropout**: El Dropout es una técnica de **regularización** que se utiliza para prevenir el **sobreajuste** (overfitting) durante el entrenamiento de una red neuronal. Durante cada iteración del entrenamiento, Dropout **desactiva aleatoriamente** un porcentaje de las neuronas, lo que obliga a la red a no depender excesivamente de ciertas neuronas y a ser más robusta. Esta técnica mejora la generalización de la red, lo que la hace funcionar mejor en datos no vistos.

Funcionamiento General de una CNN

Al pasar una imagen a través de varias capas convolucionales, la red aprende a identificar características simples como líneas y bordes. Conforme avanza a capas más profundas, las características se vuelven más abstractas,

capturando patrones más complejos como formas, texturas y, finalmente, estructuras completas como objetos.

Por ejemplo, en una red entrenada para reconocer caras, las primeras capas pueden detectar bordes o contornos, las capas intermedias pueden aprender a reconocer ojos, nariz o boca, y las últimas capas pueden identificar una cara completa.

Aplicaciones de las CNN

- **Clasificación de imágenes**: Etiquetar imágenes en distintas categorías, como identificar animales o vehículos.
- **Detección de objetos**: Identificar y localizar objetos en imágenes.
- **Reconocimiento facial**: Utilizado en sistemas de seguridad, como el desbloqueo de teléfonos móviles.

Las CNN son fundamentales en muchas aplicaciones modernas debido a su capacidad para procesar y entender datos visuales de manera eficiente y automática.

Capítulo 4

Repaso Bibliográfico

Capítulo 5

Descripción de los Algoritmos

5.1. Algoritmos Meméticos

Capítulo 6

Implementación

En este capítulo se detallan la arquitectura del sistema que se ha implementado, como la tecnología usada, diseños, etc.

El repositorio ha sido almacenado en la plataforma **Github** desde el comienzo del proyecto. De forma obvia, la herramienta usada para el control de las versiones es **Git** [].

6.1. Descripción del Sistema

La estructura del proyecto es la siguiente:

- `data` – Almacena los dataset utilizados por el proyecto.
- `docs` – Documentación del proyecto en latex.
- `img` – Imágenes generadas en el proyecto.
- `LICENSE` – Términos de distribución del proyecto.
- `README.md` – Descripción general.
- `requirements.txt` – Dependencias del proyecto.
- `results` – Resultados generados por el proyecto (fitness, tiempos, etc.).
- `scripts` – Scripts y programas secundarios para ejecutarse en el servidor GPU.
- `src` – Código fuente del proyecto.
- `utils` – Módulos y scripts de utilidad.

6.2. Herramientas y Lenguajes de Programación

El desarrollo del proyecto se ha llevado a cabo utilizando **Python 3.10** [] como lenguaje principal, debido a su versatilidad y amplia adopción en el campo del **aprendizaje profundo** y la **manipulación de datos**. Python es conocido por su facilidad de uso, extensibilidad y la gran cantidad de bibliotecas disponibles para el procesamiento de datos y la implementación de modelos de **machine learning**.

Las principales bibliotecas empleadas durante el desarrollo son las siguientes:

- **PyTorch 2.3.1** []: Biblioteca usada para implementar redes neuronales y modelos de aprendizaje profundo, con flexibilidad para experimentar y aprovechar GPU.
- **Scikit-learn 1.5.2** []: Librería para el procesamiento de datos y modelos de machine learning, que incluye herramientas para selección de características y validación cruzada.
- **Numpy 2.0.0** []: Herramienta para realizar operaciones matemáticas y de álgebra lineal, esencial para manejar grandes conjuntos de datos.
- **Polars 1.9.0** []: Alternativa más eficiente a Pandas, utilizada para gestionar grandes volúmenes de datos y mejorar el procesamiento de DataFrames.

Cada una de estas herramientas fue seleccionada por su robustez y su idoneidad para cumplir con los requisitos específicos del proyecto, facilitando tanto la implementación de los algoritmos meméticos como la reducción y el análisis de los datos utilizados en los modelos de aprendizaje profundo.

6.3. Proceso de Desarrollo

El desarrollo de este Trabajo de Fin de Grado (TFG) se inició con una cuidadosa **planificación inicial**. Se definieron **objetivos claros** y un **plan temporal** que guiaría cada una de las fases del proyecto. El primer paso consistió en llevar a cabo una **investigación preliminar** para comprender el **estado del arte** en cuanto al uso de algoritmos meméticos en el ámbito del **aprendizaje profundo**. Esta investigación incluyó la revisión de trabajos previos relevantes, así como la identificación de las **herramientas y tecnologías** más adecuadas para el desarrollo del proyecto.

Con una base teórica sólida y una planificación definida, se procedió a la **fase de desarrollo**. Esta etapa comenzó con la implementación de un **prototipo inicial**, que permitió realizar pruebas preliminares y sentar las bases del código principal. A medida que se implementaba esta primera versión, se llevó a cabo una **experimentación iterativa**, un proceso en el que se fueron incorporando **nuevos experimentos** y realizando **ajustes continuos** a partir de los resultados obtenidos. Este enfoque de **desarrollo incremental** permitió refinar el rendimiento de los algoritmos meméticos y mejorar la eficiencia de los modelos.

Finalmente, el proceso culminó con la **documentación exhaustiva** de todos los pasos realizados, así como la **preparación de la presentación** y la **defensa** del proyecto final. La documentación incluyó tanto el **análisis de los resultados experimentales** como una reflexión crítica sobre los logros y las limitaciones del TFG, asegurando que todo el proceso fuera presentado de manera clara y coherente.

Capítulo 7

Desarrollo Experimental

En este capítulo, tras haber explicado todo el conocimiento necesario para desarrollar el TFG, se explicarán todas las pruebas y mejores que se han realizado, junto a los resultados obtenidos.

Las pruebas iniciales que se plantearon fueron tomar un dataset simple para realizar las primeras pruedas, y ya cuando funcionase correctamente, probar con otro dataset más complejo o realista. Para ello se decidió usar el dataset de **RPS**.

Para obtener unos primeros resultados con este dataset, se planteó usar el modelo de **Resnet50**. Se hicieron unas pruebas con distintos porcentajes para ver cuál era el porcentaje que más merecia la pena usar.

Algoritmo	**Porcentaje Inicial**	**Duracion**	**Accuracy (Avg)**	**Precision (Avg)**	**Recall (Avg)**	**F1-score (Avg)**	**Evaluaciones Realizadas**
aleatorio	10	00:45:08	76,55	81,80	76,55	76,25	100
aleatorio	20	01:10:27	81,77	84,70	81,77	81,59	100
aleatorio	50	02:24:49	87,14	88,09	87,14	86,97	100
aleatorio	100	00:02:42	87,90	88,96	87,90	87,81	1

Tabla 7.1:

Tras realizar las pruebas iniciales, se pensó en probar con otro modelo, uno más veloz para que tardase menos tiempo en realizar pruebas y poder realizar más pruebas en menos tiempo. Para ello se decido usar **Mobilenet**, debido a que es más rápido y suele tener resultados parecidos si los Dataset no son muy grandes.

Capítulo 8

Conclusiones