
# Exploración de modelos profundos de interpolación y basados en memoria para la clasificación de imágenes naturales

**Diego Romero Iregui**

Facultad de Ciencias
Departamento de Matemáticas
Sede Bogotá, Colombia
2024

# Exploración de modelos profundos de interpolación y basados en memoria para la clasificación de imágenes naturales

**Diego Romero Iregui**

Trabajo de grado presentado como requisito para obtener el título de
**Ciencias de la Computación**

**Director:**
Prof. Dr. Fabio Augusto Gonzalez Osorio
Profesor Titular -
Facultad de Ciencias
Universidad Nacional de Colombia

- Departamento de Matemáticas
Facultad de Ciencias
Universidad Nacional de Colombia

Universidad Nacional de Colombia
Facultad de Ciencias
Departamento de Matemáticas
2024

# Declaración

Me permito afirmar que he realizado este trabajo de grado de manera autónoma y con la única ayuda de los medios permitidos y no diferentes a los mencionados el presente texto. Todos los pasajes que se han tomado de manera textual o figurativa de textos publicados y no publicados, los he reconocido en el presente trabajo. Ninguna parte del presente trabajo se ha empleado en ningún otro tipo de tesis.

Sede Bogotá., Fecha entrega

Diego Romero Iregui

# Resumen


## Exploración de modelos profundos de interpolación y basados en memoria para la clasificación de imágenes naturales

Los modelos de clasificación utilizados actualmente en múltiples aplicaciones de clasificación tienen serias limitaciones. En desempeño, todavía hay margen de mejora, los modelos del estado del arte rondan precisiones del 80 % en bases de datos con 1000 posibles predicciones. El ‘‘olvido catastrófico’’, fenómeno que impide que los modelos adquieran nuevo conocimiento sin olvidar el aprendido previamente, la alta demanda computacional, y los modelos que guardan el conocimiento implícitamente, dificultando la interacción entre el usuario y el conocimiento adquirido son otros problemas.
Una manera inteligente de atacar estos problemas es utilizar estrategias para interpolar y aproximar una solución continua utilizando un mínimo número de instancias discretas en un espacio de representación que aproxime el espacio original. Este trabajo presenta un enfoque a la interpolación que evita el fine-tuning, al utilizar la distancia a los k vecinos más cercanos de una instancia en el espacio de representación para tomar una desición de clasificación. La propuesta es una adaptación del método clásico kNN en un espacio en que los datos normalizados apoyan que la métrica del espacio es euclidiana. La propuesta fue evaluada en diferentes conjuntos de datos con hasta 1 200 000 imágenes y 1000 clases, observándose un desempeño cercano o superior al de los modelos del estado del arte, con relativamente pocos recursos computacionales, además de dar solución a los demás problemas mencionados.


**Palabras clave: Red Neuronal Profunda, Interpolación, Knn**

# Contenido

# 1 Introducción

Recientemente, el procesamiento de imágenes ha experimentado una revolución gracias al desarrollo de modelos de aprendizaje profundo. La clasificación precisa de imágenes naturales es esencial en diversas aplicaciones, desde diagnósticos médicos hasta vehículos autónomos. Sin embargo, a pesar del impacto generado por estas tecnologías emergentes, todavía es necesario resolver muchos problemas puntuales, como la variabilidad en la calidad de las imágenes y la necesidad de manejar conjuntos de datos no balanceados. En este contexto, es necesaria la exploración continua de enfoques más sofisticados.
Esta propuesta de trabajo de grado se centra en la investigación y desarrollo de modelos innovadores de clasificación de imágenes naturales. Para ello, se busca abordar la variabilidad en la calidad de las imágenes mediante la creación de representaciones intermedias para que el modelo capture información relevante incluso en condiciones subóptimas o cuando el conjunto de datos es muy pequeño.

## 1.1 Planteamiento del Problema

Los algoritmos de clasificación de imágenes han probado ser muy valiosos por sus numerosas aplicaciones. Como ya se mencionó, no se trata de un problema resuelto, a causa de obstáculos como la variabilidad de los datos. La calidad de las imágenes, los conjuntos de datos desbalanceados, la variedad de tomas de un mismo objeto, son algunos ejemplos.
A pesar de acercarse a resolver los obstáculos anteriores, los modelos de Deep Learning actuales presentan problemas adicionales.

Los modelos expuestos previamente guardan de manera implícita la información aprendida en los parámetros del mismo. Esto dificulta la interacción entre el usuario y el modelo, pues es difícil saber cómo se está utilizando la información aprendida para clasificar los datos.
Esto resulta en limitaciones. Por ejemplo, un usuario no puede extraer los parámetros que el modelo utiliza para clasificar una de las clases en específico. Si entre los datos de entrenamiento hay imágenes impertinentes, o con etiquetas erróneas, la red aprende conocimiento falso, y el usuario no tiene forma de corregir específicamente este conocimiento. Además, los usuarios no pueden evaluar la contribución de la información aprendida a los resultados.

Para intentar ‘‘afinar’’ un modelo de redes neuronales profundas ya entrenado, se ajustan los parámetros usando nuevos datos y considerando la función de costo. Desafortunadamente, al realizar este proceso el modelo olvida parte del conocimiento adquirido previamente. Esto se conoce como ‘‘Catastrophic Forgetting’’.

Se puede intentar mitigar este fenómeno, sin embargo no podrá ser eliminado por completo, a menos que se abandone la posibilidad de adquirir nuevo conocimiento.

De acuerdo con la teoría propuesta, los modelos de interpolación pueden ser tan capaces como los modelos profundos. En este trabajo queremos responder la siguiente pregunta: ¿Pueden los modelos basados en interpolación con kNN igualar o superar las redes neuronales sobre-parametrizadas?

## 1.2 Hipótesis

Con los modelos propuestos en este documento exploramos una solución a los problemas expuestos previamente. Plantemos como arquitectura general un modelo conformado por dos estructuras principales: Primero, una red neuronal convolucional profunda previamente entrenada. Se usa para procesar una imágen como lo haría habitualmente, pero en lugar de tomar los resultados de la última capa, se extraen los valores de una capa intermedia, preferiblemente cercana a la última. Así podremos crear representaciones, o ‘‘embeddings’’ de las imágenes.
Estos embeddings son luego utilizados para alimentar un modelo de kNN, que realiza la predicción de las etiquetas. Es decir, si se solicita predecir la etiqueta de una imagen, el modelo primero calcula su representación, a la cual después aplica un algoritmo kNN, determinar dentro de qué clases están las imágenes más parecidas y con esta información realizar una predicción.

Por un lado, como se mostró en el estado del arte, los modelos basados en interpolación pueden tener beneficios respecto a las redes neuronales profundas clásicas en rendimiento. Además, el modelo kNN ayuda a resolver los problemas de Catastrophic Forgetting y aprendizaje implícito mencionados anteriormente.
Dado que en este modelo el aprendizaje consiste en alimentar de más representaciones el kNN, no se perderá el conocimiento adquirido nuevamente, por el contrario la base de datos de representaciones podría expandirse indefinidamente. Y dado que el conocimiento adquirido está codificado en las representaciones, es mucho más fácil para el usuario entender cómo se está clasificando. Gracias a esto resulta posible tomar subconjuntos de las representaciones, evaluar la utilidad de la información aprendida y eliminar representaciones que no sean útiles.

## 1.3 Objetivos

### 1.3.1 Objetivo General

Desarrollar y evaluar un modelo basado en interpolación para la clasificación de imágenes.

### 1.3.2 Objetivos Específicos

- Desarrollar un modelo de clasificación de vecino más cercano para grandes volúmenes de datos combinando un modelo de red neuronal profunda pre-entrenado y una base de datos de vectores
- Evaluar el desempeño del modelo en el conjunto de datos usado para pre-entrenar el modelo profundo (por ejemplo ImageNet).
- Evaluar el desempeño del modelo en otros conjuntos de imágenes diferentes al conjunto de entrenamiento y comparar el desempeño con estrategias de "transfer learning" y "fine tuning".

## 1.4 Contribuciones

Se experimentó con múltiples diseños de modelos y se probaron en diferentes conjuntos de datos. Además de el documento del trabajo de investigación, con el detalle de los métodos usados, el reporte de los resultados, y las conclusiones, se entregan un total de 10 archivos de cuaderno, cada uno con una implementación y dataset diferentes, así como el código utilizado para separar los datos.

# 2 Marco teórico

## 2.1 Conceptos

### 2.1.1 Deep Learning

Las redes neuronales son un modelo computacional, que consiste en una red de nodos interconectados, llamados neuronas artificiales o unidades, que trabajan en conjunto para resolver tareas específicas, como reconocimiento de patrones, clasificación de datos o predicción.

En un nivel básico, una red neuronal consta de capas de neuronas, donde cada neurona está conectada a todas las neuronas de la capa siguiente. Estas conexiones se representan por pesos, que determinan la fuerza y la dirección de la influencia que una neurona tiene sobre otra. La información fluye a través de la red desde las neuronas de entrada, a través de una o más capas ocultas, hasta las neuronas de salida, donde se produce el resultado deseado.

Durante el entrenamiento, la red neuronal ajusta automáticamente los pesos de sus conexiones en función de los datos de entrada y las salidas deseadas, utilizando algoritmos de aprendizaje. Este proceso permite a la red adaptarse y mejorar su capacidad para realizar la tarea específica para la que ha sido diseñada.

Los modelos de "Deep Learning"son redes neuronales con múltiples capas ocultas entre la capa de entrada y la capa de salida, con muchos parámetros que se entrenan con grandes volúmenes de datos. Estas capas ocultas permiten a la red aprender representaciones de datos a diferentes niveles de abstracción, lo que mejora su capacidad para realizar tareas complejas de clasificación, regresión o generación de datos. A medida que los datos pasan a través de las capas, se transforman mediante una serie de funciones no lineales, lo que permite a la red capturar relaciones complejas y patrones intrincados en los datos.

El entrenamiento de estas redes se realiza generalmente mediante un algoritmo de retropropagación, que ajusta los pesos de las conexiones neuronales para minimizar una función de pérdida. Este proceso requiere una gran cantidad de datos para generalizar efectivamente y evitar el sobreajuste. Además, el uso de técnicas como la regularización, el dropout y la normalización de lotes ayuda a mejorar el rendimiento y la estabilidad del modelo durante el entrenamiento.

El éxito de los modelos de Deep Learning en diversas aplicaciones, como el reconocimiento de voz, la visión por computadora y el procesamiento del lenguaje natural, se debe en gran parte a su capacidad para aprender automáticamente características relevantes de los datos, eliminando la necesidad de extracción manual de características. Esto ha llevado a avances significativos en campos como la inteligencia artificial y el aprendizaje automático, abriendo nuevas posibilidades para la investigación y el desarrollo de aplicaciones innovadoras.

### 2.1.2 Modelos de Deep Learning para clasificación de imágenes

La capacidad de asignar automáticamente etiquetas o categorías a imágenes permite una organización eficiente de grandes volúmenes de datos visuales, facilitando la toma de decisiones informadas y la automatización de procesos. En este contexto, el deep learning ha emergido como un enfoque poderoso y versátil para abordar el desafío de la clasificación de imágenes. Al entrenar modelos de redes neuronales profundas en conjuntos de datos masivos, el deep learning permite aprender automáticamente las características relevantes de las imágenes, adaptándose así a una amplia variedad de dominios y mejorando continuamente su precisión y robustez. Estos son algunos de los diferentes enfoques para la clasificación de imágenes usando deep learning:

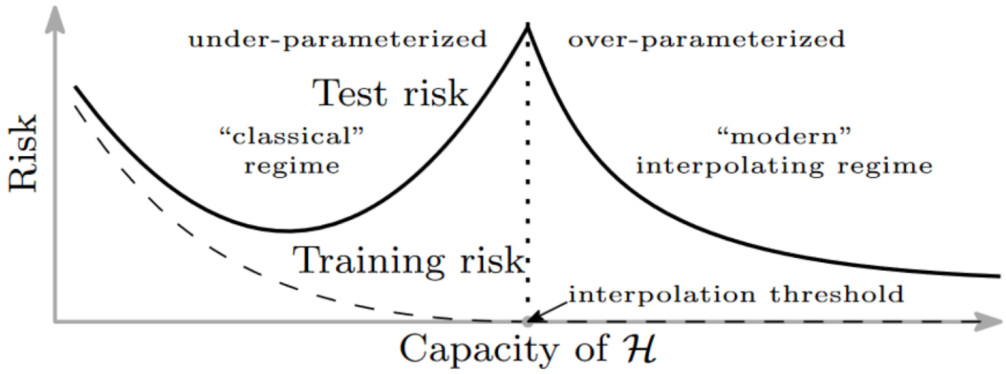
- **Redes Neuronales Convolucionales (CNN):** Las CNN son el enfoque dominante para la clasificación de imágenes. Utilizan capas convolucionales para extraer características jerárquicas de las imágenes, capas de agrupamiento (pooling) para reducir la dimensionalidad y capas totalmente conectadas para la clasificación final. [15]
- **Transfer Learning:** Aprovecha modelos pre-entrenados en conjuntos de datos masivos, como ImageNet, y los adapta a tareas específicas con conjuntos de datos más pequeños. Reduce el tiempo y los recursos necesarios para entrenar un modelo desde cero y puede mejorar el rendimiento cuando los datos de entrenamiento son limitados. [16]
- **Redes Neuronales Recurrentes (RNN):** Útiles para clasificar secuencias de imágenes o videos. Las LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit) son variantes populares de las RNN que pueden capturar relaciones temporales en los datos. [17]
- **Redes Neuronales Siamesas:** Utilizadas para problemas de comparación o similitud entre imágenes. Comparten pesos entre dos redes idénticas para generar representaciones similares para imágenes similares, útiles en tareas como reconocimiento de rostros o búsqueda de imágenes similares. [18]
- **Redes Generativas Adversarias (GAN):** No se utilizan directamente para clasificación, pero pueden ser útiles para aumentar datos de entrenamiento o generar ejemplos de clases minoritarias. Generan imágenes realistas que pueden utilizarse en combinación con imágenes reales para entrenar modelos de clasificación. [19]
- **Atención visual:** Mejora la capacidad de las CNN para enfocarse en regiones importantes de una imagen, permitiendo que el modelo "preste atención.ª partes específicas de la imagen durante el proceso de clasificación. [20]
- **Transformers:** Originalmente diseñados para tareas de procesamiento de lenguaje natural, pero han demostrado eficacia en el procesamiento de imágenes. Utilizan mecanismos de autoatención para captar relaciones espaciales entre píxeles. [21]

### 2.1.3 Aprendizaje basado en interpolación

La teoría matemática del aprendizaje estadístico presenta dificultades para dar una explicación convincente a sus éxitos, así como para ayudar al diseño de nuevos algoritmos o la mejoría de las arquitecturas neuronales. La interpolación, la idea de encajar los datos de entrenamiento de manera perfecta, es menos dependiente de estadística pero contraintuitiva a causa del ruido en los datos y el problema de “overfitting”. La sobreparametrización en los modelos es aquello que permite la interpolación y da flexibilidad al seleccionar el correcto.

Se solía creer que después de un punto, al aumentar los parámetros en un modelo, éste se alejaba cada vez más de la cantidad óptima de error. Sin embargo, se empieza a considerar que al aumentar cada vez más

los parámetros, se llega un punto donde el error vuelve a disminuir. Por esto, es importante considerar los modelos de sobre-parametrización, pues pueden ser la clave para más precisión en las predicciones.




Cuando la capacidad de una clase de hipótesis H está por debajo del umbral de interpolación, es decir, no es suficiente para ajustar datos arbitrarios, los predictores aprendidos siguen la clásica curva en forma de U [1]. La forma de la curva de generalización experimenta un cambio cualitativo cuando la capacidad de H supera el umbral de interpolación, es decir, se vuelve lo suficientemente grande como para interpolar los datos. Aunque los predictores en el umbral de interpolación suelen tener un alto riesgo, aumentar aún más el número de parámetros (capacidad de H) conduce a una mejor generalización. El patrón de doble descenso ha sido demostrado empíricamente para una amplia gama de conjuntos de datos y algoritmos, incluyendo redes neuronales profundas modernas [6,7,8] y se observó anteriormente para modelos lineales [9]. El régimen "moderno"de la curva, el fenómeno de que un gran número de parámetros a menudo no llevan al sobreajuste, ha sido observado históricamente en boosting [10, 11] y bosques aleatorios, incluyendo bosques aleatorios interpolantes [12], así como en redes neuronales [13, 14].

### 2.1.4 k-Nearest-Neighbors (kNN)

Los modelos kNN parten de la premisa de que los datos similares suelen estar cercanos unos o otros en un espacio donde se encuentran las representaciones de sus características. Estos modelos se entrenan al recibir un conjunto de representaciones para poder realizar comparaciones. Al recibir un nuevo dato, calculan la distancia de su representación a la de las demás representaciones guardadas en el modelo. Se consideran las k menores distancias, y se hace la predicción calculando cuál es la etiqueta más frecuente entre esos k vecinos más cercanos.

## 2.2 Estado del arte

Aunque este problema ha sido reformulado en los últimos tiempos con la explosión de aplicaciones del aprendizaje profundo y la necesidad de generalizar las soluciones encontradas, la interpolación de datos es un problema de vieja data que ha sido aproximado de muchas maneras, desde aproximaciones lineales hasta los spline. Sin embargo, la mayoría de estas soluciones serían impracticables en el contexto del aprendizaje de máquina por la cantidad de datos y de parámetros que se necesitan en un problema real.

Diferentes aproximaciones han intentado resolver este problema. Papernot y MacDaniel [2], introducen los "Deep k-Nearest Neighbors" (DkNN), un clasificador híbrido que combina el algoritmo de k-vecinos más cercanos con representaciones de los datos aprendidos por cada capa del DNN. Una entrada de prueba es comparada a sus puntos de entrenamiento vecinos de acuerdo a la distancia que los separa en las representaciones.
Chawin Sitawarin y David Wagner [3] proponen un ataque heurístico con descenso de gradiente para encontrar ejemplos adversos para clasificadores kNN y después aplicarlo para abordar también la defensa DkNN.
Liron Bergman, Niv Cohen y Yedid Hoshen, [4] muestran que la aproximación basada en kNN experimentalmente supera los métodos de aprendizaje autosupervisado en precisión, tiempo de entrenamiento, y robustez contra el ruido, todo realizando menos suposiciones sobre distribución de imágenes.
Yiyou Sun, Yifei Ming, Xiaojin Zhu y Yixuan Li [5] exploran la eficacia de distancia no paramétrica de kNN para detección "Out of Distribution" (ODD). A diferencia de trabajos anteriores, su método supone una distribución, lo cual provee más flexibilidad y generalidad.

# 3 Métodos

## 3.1 Modelos de clasificación de imágenes

Keras Applications es un módulo de la librería Keras que incluye modelos de redes neuronales profundas previamente entrenados. Estos modelos se entrenaron con datos de Imagenet, una base de datos con 1000 etiquetas, por lo que la última capa de los modelos es un vector de tamaño 1000, donde cada valor indica la intensidad de activación del nodo correspondiente, que a su vez lleva asociada una etiqueta. Se implementaron algunos de estos modelos, y se verificó el desempeño de cada uno. ResNet50, MobileNetV2 y ConvNeXt-Tiny serían los utilizados posteriormente. Específicamente se probaron los modelos siguientes:

- VGG16
- VGG19
- Resnet50
- Resnet50V2
- MobileNet
- MobileNetV2
- ConvNeXtTiny
- ConvNeXtSmall
- ConvNeXtBase

### 3.1.1 MobileNetV2

MobileNet-v2 es una red neuronal profunda convolucional con 53 capas de profundidad basada en el uso de bloques residuales invertidos y convoluciones por profundidad, que permiten una notable reducción en la cantidad de parámetros y operaciones. Los bloques residuales invertidos incluyen una fase de expansión, una convolución depthwise y una fase de proyección, optimizando la eficiencia al separar las operaciones espaciales y de canal. La red preentrenada puede clasificar imágenes de tamaño 224x224 en 1000 categorías de objetos. En el diagrama siguiente puede verse la arquitectura de la red:

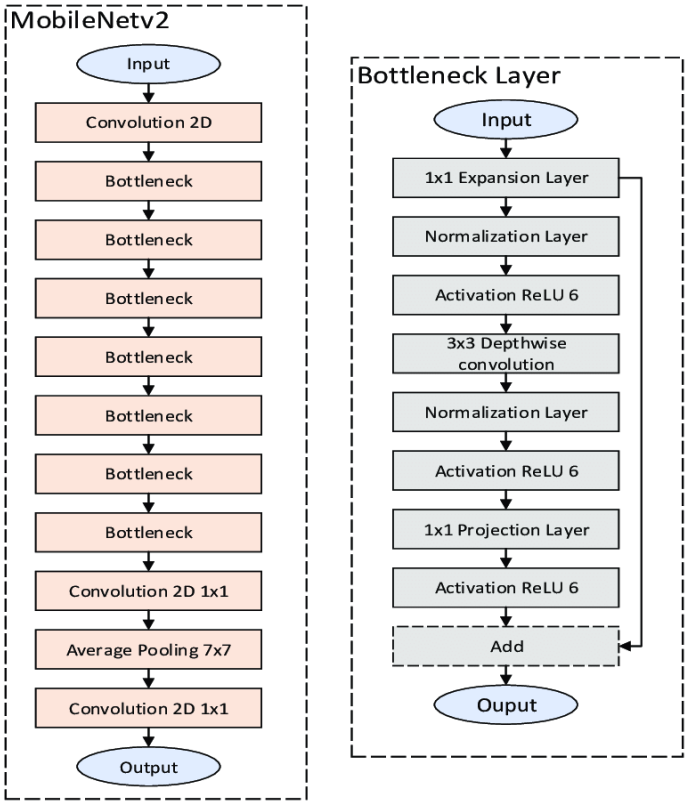



**Figura 3-1**: Diagrama de Arquitectura de MobileNetV2

### 3.1.2 ResNet50

ResNet50 es una arquitectura de red neuronal profunda diseñada para abordar el problema del desvanecimiento del gradiente, con el uso de bloques residuales, que es común en redes con muchas capas. El diagrama siguiente muestra la arquitectura del modelo:

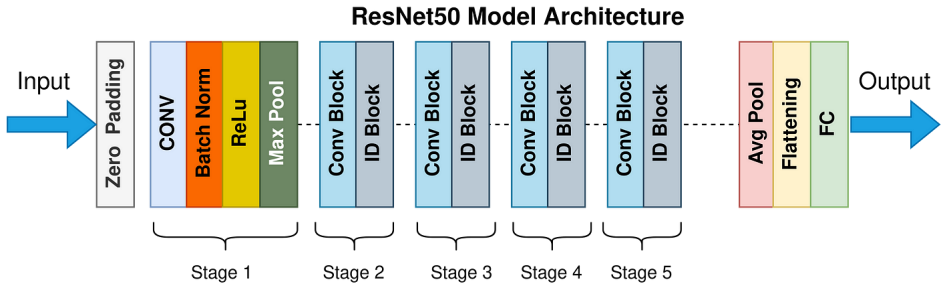



**Figura 3-2**: Diagrama de Arquitectura de ResNet50

ID Block: consiste de tres capas convolucionales, seguidas por funciones de "batch normalization" y activación

ReLU. la entreda se suma con la salida de la tercera capa convolucional.

### 3.1.3 ConvNeXt-Tiny

ConvNeXt-Tiny es una arquitectura de red neuronal convolucional que se basa en las redes convolucionales clásicas y se inspira en las innovaciones recientes, como los transformers, integrando elementos modernos que mejoran su rendimiento.

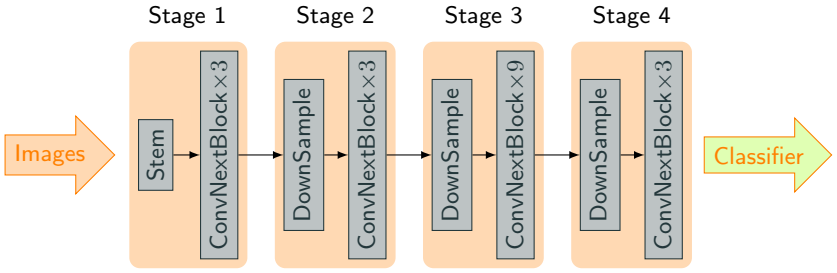



**Figura 3-3**: Diagrama de Arquitectura de ConvNeXt-Tiny, Parte 1

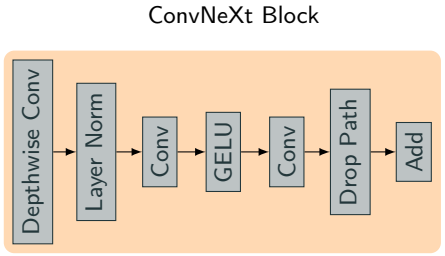



**Figura 3-4**: Diagrama de Arquitectura de ConvNeXt-Tiny, Parte 2

### 3.1.4 Red Siamesa

Las redes siamesas son redes neuronales diseñadas para comparar imágenes y modificar la generación de representaciones según lo deseado. Para este proyecto se utilizó la red siamesa de los ejemplos de visión por computadora de Keras.
Esta consiste en tres subredes ResNet50, que procesan tres entradas diferentes, simultáneamente, y producen vectores de características (embeddings), al limitar la ejecución de Resnet hasta la capa ‘‘conv5_block1_out’’ y tomar el estado de la misma como embedding.
Las tres entradas son: una imagen ‘‘ancla’’, un ejemplo positivo del ancla, es decir una imagen con la misma etiqueta, y un ejemplo negativo del ancla, una imagen con etiqueta distinta.

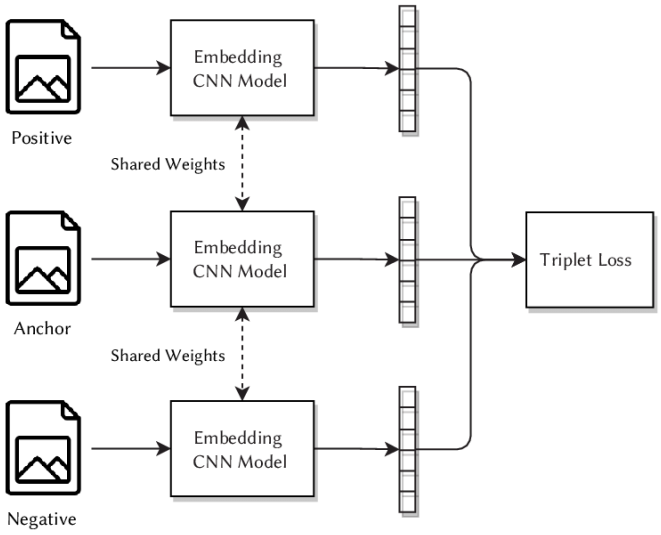



**Figura 3-5**: Diagrama de Red Siamesa

Al haber computado los tres embeddings, se calcula la función triplet loss, definida como:

$$\text{Triplet Loss} = max(0, d(a,p) - d(a,n) + \text{margen})$$

donde d(a,p) es la similitud coseno entre el ancla y el positivo, d(a,n) es la similitud coseno entre el ancla y el negativo, y margen es un valor positivo que establece un límite mínimo para la diferencia de distancias. Finalmente, los pesos de las subredes se actualizan a través de retropropagación para mejorar la capacidad de la red.
Para el uso de la red siamesa, se separaban los datos de entrenamiento en parejas de ancla y ejemplo positivo, asegurando que pertenecieran a la misma clase. Luego, al llamar la red, se insertaba la pareja junto con un ejemplo negativo escogido al azar entre otros elementos del dataset.

### 3.1.5 Scikit-learn kNN

El modelo k-Nearest Neighbors (kNN) es un método para problemas de clasificación, que predice la etiqueta de un dato basándose en la etiqueta de los datos más cercanos en un espacio de características. Para este proyecto, se utilizó la sub-librería KNeighborsClassifier, de Scikit-Learn.
Como primer paso, se debe alimentar el modelo con suficientes datos de entrenamiento para que este tenga elementos con los que hacer comparaciones y determinar similitudes. Al recibir una entrada, el modelo identifica los k datos de entrenamiento más cercanos según la distancia euclidiana. La predicción del modelo para la nueva instancia se determina por la clase mayoritaria entre los k vecinos más cercanos.

La figura anterior ilustra la idea detrás del modelo, y destaca la influencia que puede tener el valor k sobre la predicción final.

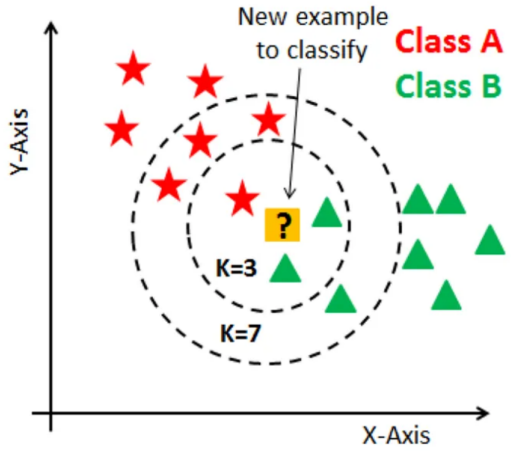
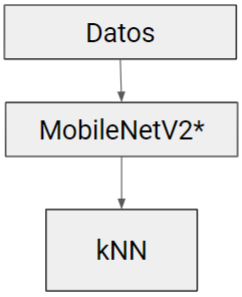



**Figura 3-6**: Diagrama de kNN

## 3.2 Modelos híbridos CNN-KNN

### 3.2.1 MobileNetV2 y kNN

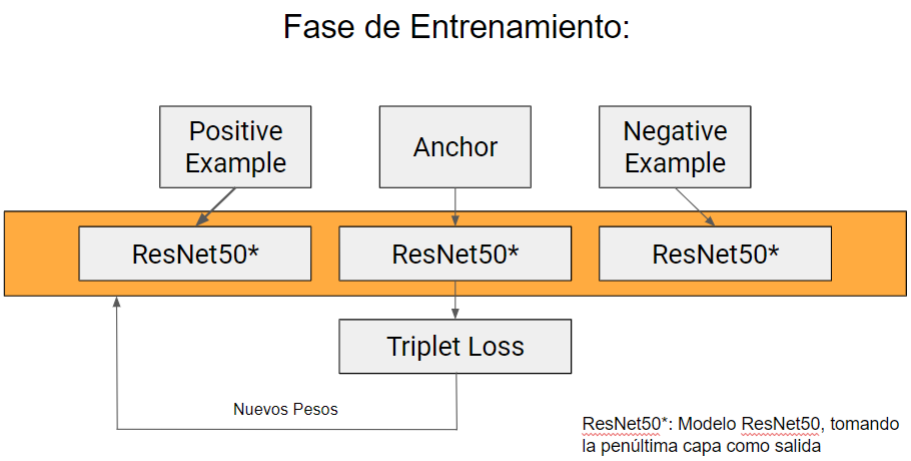
Una vez implementado el llamado a los métodos, se usó la librería Keras para tomar una capa intermedia como salida, y así generar embeddings de los datos. Luego estas representaciones entrenan un modelo kNN que calcula distancias tomando como métrica la distancia euclidiana. Si después se desea clasificar una imagen con este nuevo modelo, primero se calcula su embedding, que luego se inserta en el kNN para que encuentre los vecinos más cercanos y realice la predicción.
Se realizó un primer experimento con MobileNetV2, utilizando Imagenet. Con la penúltima capa se generaron embeddings de 20000 imágenes de Imagenet y con estos se creó un modelo kNN. En este experimento y en los siguientes se varió el valor de k y se tomó el valor que mostrara el mejor desempeño.
En múltiples instancias se implementó un modelo con la siguiente arquitectura:



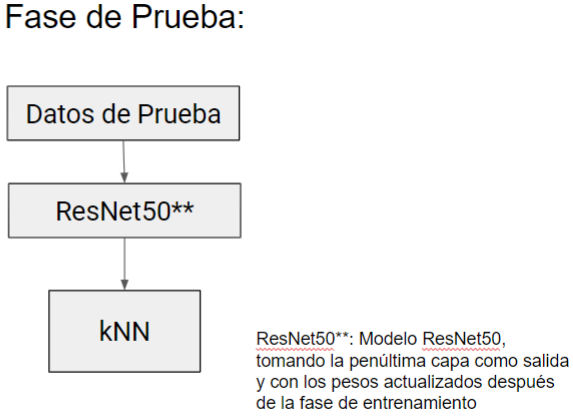

MobileNetV2*: Modelo MobileNetV2,
tomando la penúltima capa como salida

### 3.2.2 MobileNetV2, Red Siamesa y kNN

Para intentar mejorar los resultados, se implementa una arquitectura de red siamesa. Esta modifica los pesos de ResNet50 con el propósito de generar embeddings que resulten en mejor desempeño, más precisamente tal que la distancia entre embeddings de una misma clase disminuya, y la distancia inter-clase aumente. La red Siamesa calcula estas distancias entre embeddings con la distancia coseno, para mayor rendimiento acorde a la literatura.
Luego, análogo al experimento anterior, los nuevos embeddings alimentan el modelo kNN que después realizará las predicciones. Sin embargo, como la red Siamesa usa distancia coseno y el kNN distancia euclidiana, se normalizan los datos de salida de la red para mantener la coherencia con el cambio de métrica. En la etapa de entrenamiento, este nuevo modelo opera de la manera siguiente:




Con la arquitectura de red siamesa, se calculan los embeddings de las tripletas, y se ajustan los pesos de ResNet50 para disminuir la varianza intra-clase y aumentar la varianza inter-clase. Después, en la fase de prueba se puede usar ResNet50 con los nuevos pesos, análogamente al experimento anterior.

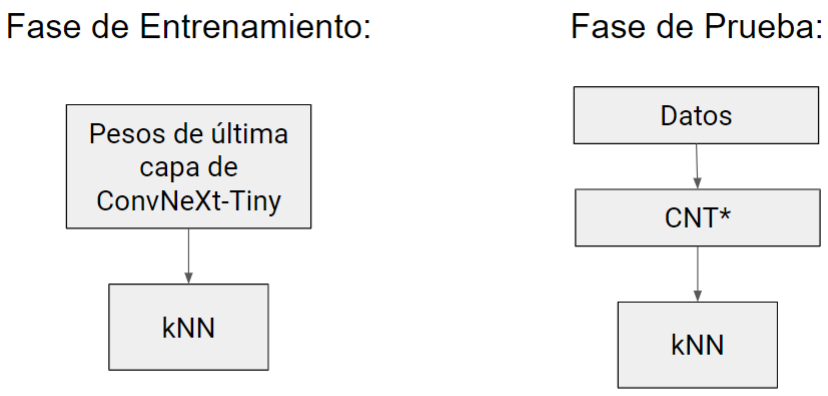


### 3.2.3 ConvNeXt-Tiny y kNN

Se retornó a la propuesta inicial, un modelo que crea embeddings con una red neuronal profunda que luego entrenan un kNN. La arquitectura sigue siendo la misma, sin embargo esta vez se utilizó una red neuronal más sofisticada, ConvNeXtTiny.

### 3.2.4 kNN entrenado con pesos de una capa

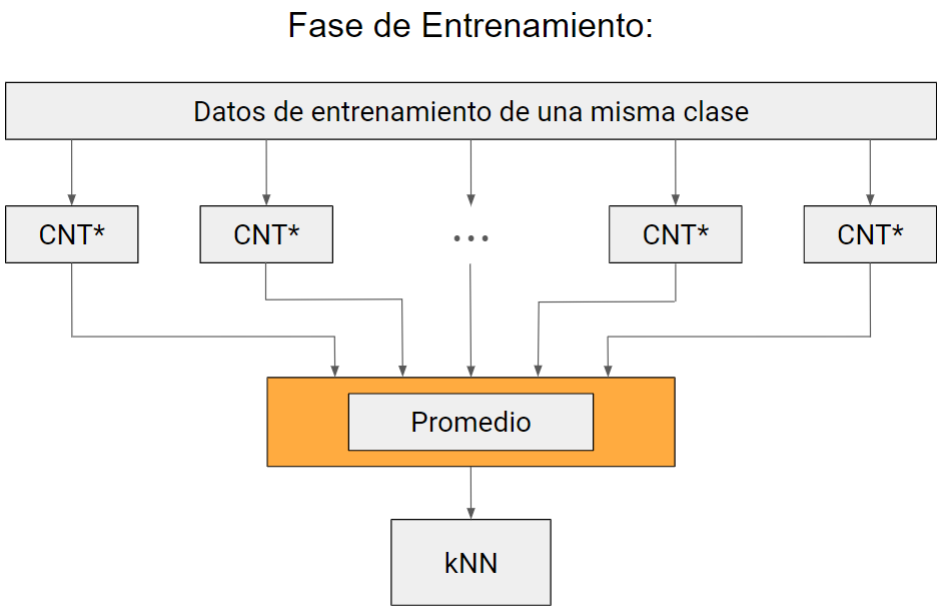
Finalmente se experimentó con dos maneras adicionales de hacer kNN. La primera es este modelo, que entrena el kNN únicamente con 1000 elementos, tomados de los pesos la última capa de ConvNeXt-Tiny. Para realizar la predicción, el modelo crea un embedding de la imagen entrante con la penúltima capa de ConvNeXt-Tiny que luego procesa el kNN.




CNT*: Modelo ConvNeXt-Tiny, tomando la penúltima capa como salida

### 3.2.5 kNN promediando los vectores de características

En este experimento, en vez de entrenar el kNN con los embeddings de todos los datos de entrenamiento, los embeddings se separan por clases, y promedian para crear un único representante por clase. Estos representantes son los que entrenan el kNN. La fase de prueba seguirá siendo la misma, se calcula el embedding y este entra en el kNN, y este diagrama muestra la fase de entrenamiento:




## 3.3 Bases de Datos

### 3.3.1 Imagenet

Imagenet Large Scale Visual Recognition Challenge 2012 (o ILSVRC 2012), conocida comúnmente como Imagenet es una base de datos de imágenes creada para su uso en el ámbito del aprendizaje de maquina y algoritmos de clasificación, específicamente para la competencia anual de 2012 que forma parte del proyecto ImageNet. El dataset tiene 1000 posibles etiquetas, o ‘‘clases’’ para cada imagen, y cada imagen tiene una única etiqueta. Las imágenes, de dimensiones variadas, están divididas entre 3 subconjuntos: Entrenamiento, Validación, y Prueba. El dataset de entrenamiento cuenta con 1 281 168 imágenes, el de validación con 50 000, y el de prueba cuenta con 100 000.

### 3.3.2 Cifar-100

Cifar 100 también es un dataset con imágenes diseñado para la investigación por computadora, aunque menos exigente computacionalmente. Las imágenes son de tamaño 32x32 y tienen 100 posibles clases. Hay en total 60 000 imágenes, 50 000 imágenes de entrenamiento y 10 000 imágenes de prueba.

### 3.3.3 Cifar-10

Similarmente, las imágenes de Cifar-10 son de tamaño 32x32, hay 50 000 destinadas al entrenamiento y 10 000 destinadas a las pruebas, pero sólo 10 etiquetas posibles para cada imagen.

## 3.4 Medidas de desempeño

Para medir el desempeño de los nuevo modelos se usaron las medidas top-1 y top-5, que pueden entenderse de la siguiente manera:
Nótese que en los modelos kNN es posible obtener más de una predicción, y ordenarlas por probabilidad. Entre los $k$ vecinos más cercanos, la clase mayoritaria será la predicción más probable, luego la segunda clase mayoritaria será la segunda predicción, y así sucesivamente.
Top-1 mide, entre los datos de prueba, cuántas veces la primera predicción acertó, y top-5 mide cuántas veces la etiqueta correcta se encontraba entre las 5 primeras predicciones.

# 4 Resultados

## 4.1 MobileNetV2 y kNN

Con el valor de k que conlleva al mejor desempeño se obtuvieron los siguientes valores de desempeño:

| Top-1 | Top-5 |
|---|---|
| 5.1 % | 11.5 % |

**Tabla 4-1**: Desempeño Top-1 de los datasets con el modelo híbrido CNN, siamesa, KNN

Los resultados eran aún muy lejanos a lo deseado, puesto que los modelos que se utilizaron (MobileNet, ResNet, ...) tienen rendimientos alrededor de 80 % en top-1 según la documentación. La razón más probable de esta diferencia es que lo modelos utilizados no fueran compatibles con la idea de alimentar un kNN con su última capa, puede ser por la manera en la que codifican el conocimiento aprendido en el entrenamiento o los tamaños de las capas.

## 4.2 MobileNetV2, Red Siamesa y kNN

Esta vez se hicieron pruebas con diferentes datasets, y se repartieron los datos de entrenamiento de diferentes maneras. La tabla siguiente resume las pruebas:

| | Imagenet | Cifar 100 | Cifar 10 |
|---|---|---|---|
| Imágenes destinadas a entrenamiento de red siamesa | 65000 | 20000 | 20000 |
| Imágenes destinadas a entrenamiento de kNN | 631167 | 25000 | 25000 |
| Imágenes para pruebas | 50000 | 5000 | 5000 |
| Precisión Top 1 | 2.3 % | 8.6 % | 41 % |
| Precisión Top 5 | 7.5 % | 26 % | 88 % |

**Tabla 4-2**: Desempeño de los datasets con el modelo híbrido CNN, siamesa, KNN

Puede verse una gran diferencia en el desempeño entre los 3 datasets utilizados. Véase enseguida las curvas de pérdida generadas durante el entrenamiento de la red siamesa en los 3 casos:

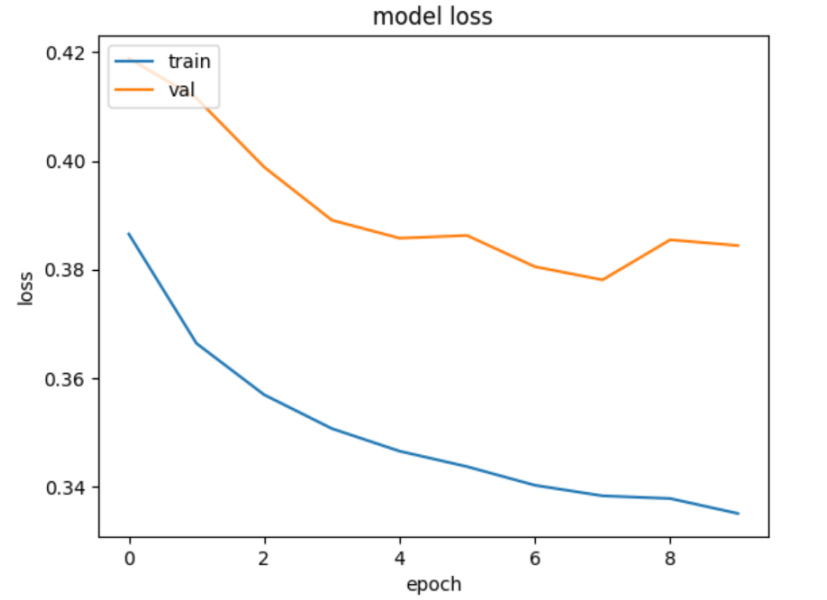



**Figura 4-1**: Pérdida a través del entrenamiento con Imagenet

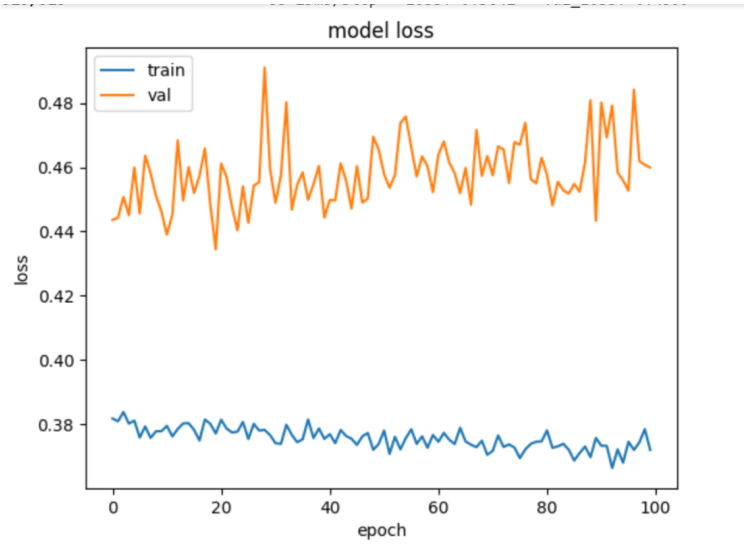



**Figura 4-2**: Pérdida a través del entrenamiento con Cifar 100

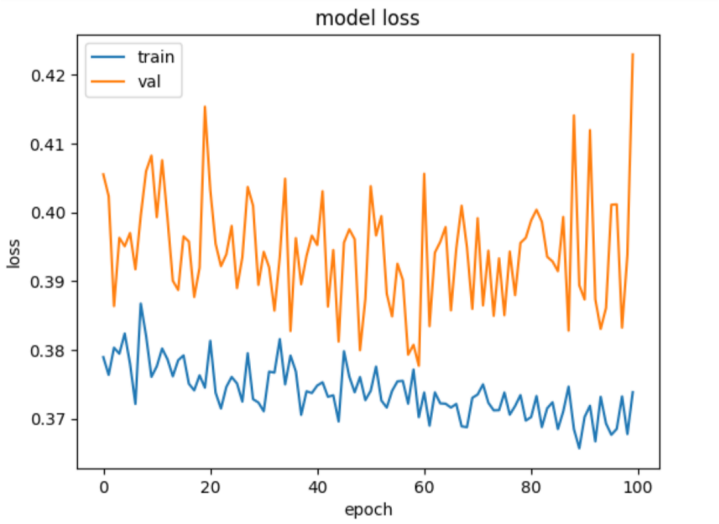



**Figura 4-3**: Pérdida a través del entrenamiento con Cifar 10

Como puede verse, aunque la pérdida en los datos de entrenamiento (azul) disminuya, parece que, desafortunadamente, después de un punto esto no conlleva a la disminución de la pérdida en los datos de validación (naranja). Puede notarse además que para Imagenet se computaron menos epochs, dado el tamaño del dataset, lo que también se ve reflejado en el resultado final.
En todo caso, a pesar de que esta aproximación parece ser eficiente para datasets con pocas clases como cifar 10 y cifar 100, los resulados en imagenet seguían sin alcanzar los niveles deseados. Además de la mayor cantidad de clases, Imagenet cuenta con muchas más imágenes y de mayor resolución, lo que obligó a computar menos epochs, lo cual pudo influir el resultado.

## 4.3 ConvNeXt-Tiny y kNN

Con este cambio de red neuronal profunda los resultados mejoraron considerablemente. Se obtuvo una certeza de 77 % en top-1, usando 50000 imágenes de prueba tomadas del subconjunto de validación de Imagenet, que son diferentes a las que fueron utilizadas para entrenar ConvNeXtTiny. Ese resultado se obtuvo para k=5. Posteriormente se probaron más valores de k pero el desempeño no variaba significativamente.

## 4.4 kNN entrenado con pesos de una capa

El modelo que crea 1000 embeddings a partir de los pesos de la última capa de ConvNextTiny obtuvo una certeza de 90 % en top-1 con datos de un subconjunto de Imagenet, ‘‘imagewoof’’.

## 4.5 kNN promediando los vectores de características

El segundo método, tomar los embeddings producidos por la última capa de ConvNeXtTiny, promediar los que pertenezcan a una misma clase y entrenar el modelo kNN con estos promedios, se probó los mismos 50 000 datos de validación de Imagenet, y la certeza obtenida pasó de 77 % a 81 % en top 1.
Usar ConvNeXtTiny resultó en una gran mejora respecto a los experimentos previos, y promediar los embeddings para obtener un representante por clase lo fue aún más. La diferencia está en que al promediar, se elimina parte del ruido que podría estar afectando negativamente los resultados. Cabe destacar que según la documentación ConvNeXTiny tiene una certeza de $81,3$ % en top-1. Esto quiere decir que con el último experimento casi se logra empatar el desempeño de la red original, alimentando el kNN con relativamente pocos datos. Esto sugiere la validez de las hipótesis iniciales sobre los modelos basados en interpolación.

## 4.6 Progresión del desempeño con Imagenet

El más desafiante de los datasets utilizados claramente es ImageNet, por su número de clases y tamaño de sus imágenes. En esta tabla podemos ver un resumen de cómo evolucionó el desempeño de las predicciones de sus datos a través de los experimentos del proyecto.

| | MobileNetV2 y kNN | MobileNetV2 Siamés y kNN | ConvNeXt-Tiny y kNN | ConvNeXt-Tiny y kNN con embeddings promedio |
|---|---|---|---|---|
| Desempeño Top-1 con Imagenet con el k óptimo | 5.1 % | 2.3 % | 77 % | 81 % |

**Tabla 4-3**: Desempeño Top-1 de los datasets con el modelo híbrido CNN, siamesa, KNN

## 4.7 Productos

Se crearon a lo largo del periodo del proyecto 10 notebooks, listados enseguida:

- **Text_Images_Folders_editing.ipynb**:
  Notebook utilizado para separar los datos en diferentes folders y crear las listas con las etiquetas de los elementos.
- **ImageNet_FirstTest.ipynb**:
  Primer intento donde se conjuntan MobileNetV2 y kNN
- **ImageNet_Siamese.ipynb**:
  Experimento de mejorar embeddings con una red siamesa para mejorar el rendimiento de kNN con el dataset ImageNet
- **Cifar 100_Siamese.ipynb**:
  Mismo experimento con el dataset Cifar 100
- **Cifar10_Siamese.ipynb**:
  Mismo experimento con el dataset Cifar 10
- **val_imagenet_embeddings_knn.ipynb**:
  Se conjuntan ConvNeXt-Tiny y kNN, y se prueba con los 50 000 datos de validación de Imagenet. El kNN sólo considera k=5
- **val_imagenet_embeddings_many_knn.ipynb**:
  Mismo experimento pero considerando más valores de k
- **imagewoof_weights_knn.ipynb**:
  kNN entrenado únicamente con los pesos de la última capa de ConvNeXt-Tiny en el sub-dataset imagewoof.
- **half_imagenet_weights_knn.ipynb**:
  Mismo experimento probando con la mitad de los datos de entrenamiento de ImageNet
- **val_imagenet_averageEmbeddings_knn.ipynb**:
  Se entrena el kNN con el promedio de las representaciones de cada clase, y se prueba con los datos de validación de ImageNet.

Los notebooks están disponibles en el enlace de Github en el apéndice

# 5 Conclusiones

En conclusión, la propuesta de modelo ha demostrado ser una estrategia eficaz y robusta, con una construcción adecuada. En un inicio se tuvo resultados muy bajos (Tabla 4-1), sería correcto concluir que la implementación con MobileNetV2 no es viable. El intento posterior, que procuraba traer un mejor desempeño, fue un paso en la dirección contraria. La Tabla 4-2 muestra que en bases de datos con pocas clases puede ser un enfoque potencial, sin embargo en Imagenet entregó resultados peores a los obtenidos previamente. Esto pudo ser porque la arquitectura de red siamesa causó un ''over-fitting'', sería pertinente en un futuro proyecto volver a abordar este diseño de modelo con más recursos computacionales, o diferentes redes neuronales profundas. Finalmente, al realizar el mismo experimento con ConvNeXt-Tiny los resultados aumentaron de golpe. Sería también interesante investigar en un futuro exactamente por qué se evidencia una diferencia tan grande. En todo caso, se probó el potencial de esta aproximación al lograr resultados de la magnitud de los modelos del estado del arte, (Tabla 4-3), con relativamente pocos recursos computacionales.
Utilizar una red neuronal profunda preentrenada para extraer características de las imágenes permitió capturar representaciones ricas y significativas, optimizando así el rendimiento del modelo kNN. Esta combinación aprovechó la capacidad de las redes neuronales profundas para aprender características complejas y del kNN para resolver los problemas de Catastrophic Forgetting y datos implícitos, además de, como ya se mencionó, sugerir que podría alcanzarse un mejor desempeño con el poder computacional necesario. En general, los resultados obtenidos validan la viabilidad de esta metodología, y evidencian su potencial para aplicaciones en diversos dominios donde la clasificación de imágenes es crucial.

# Referencias Bibliográficas

**01. Belkin, Mikhail**: , 2021; Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation.

**02. Bergman, Liron, et al**: , 2020; Deep Nearest Neighbor Anomaly Detection.

**03. Papernot, Nicolas, Patrick McDaniel**: , 2018; Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning.

**04. Sitawarin, Chawin, David Wagner**: , 2019; On the Robustness of Deep K-Nearest Neighbors.

**05. Sun, Yiyou, et al.**: , 2022; Out-of-Distribution Detection with Deep Nearest Neighbors.

**06. Belkin, Mikhail, et al.**: , 2019; Reconcil- ing modern machine-learning practice and the classical bias–variance trade- of.

**07. Nikkaran, Preetum, et al.**: , 2019; Deep double descent: Where bigger models and more data hur.

**08. Spigler, et al.**: , 2019; A jamming transition from under- to over-parameterization affects generalization in deep learning.

**09. Loog, Marco, et al.**: , 2020; A brief prehistory of double descent.

**10. Schapire, Robert E., et al.**: , 1998; Boosting the margin: a new explanation for the ef ectiveness of voting methods.

**11. Wyner, Abraham J., et al.**: , 2017; Ex- plaining the success of adaboost and random forests as interpolating classifiers.

**12. Cutler, Adele, Guohua Zhao**: , 2001; Pert-perfect random tree ensembles.

**13. Breiman, Leo**: , 1995; Reflections after refereeing papers for nips.

**14. Neyshabur, Behman, et al.**: , 2015; In search of the real inductive bias: On the role of implicit regularization in deep learning.

**15. LeCun, Bengio, Hinton**: , 2015; Deep learning.

**16. Yosinski, Clune, Bengio, Lipson**: , 2014; How transferable are features in deep neural networks?

**17. Hochreiter, Schmidthuber**: , 1997; Long short-term memory.

**18. Bentz, Bottou, Guyon, LeCun, Moore, Säckinger**: , 1994; Signature verification using a "Siamese"time delay neural network.

**19. Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Bengio**: , 2014; Generative adversarial nets.

**20. Mnih, Heess, Graves, et al.**: , 2014; Recurrent Models of Visual Attention.

**21. Dosovitskiy, Beyer, Kolesnikov, Weissenborn, Zhai, Unterthiner, Houlsby**: , 2020; An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

# A Apéndice

## Fuentes de Figuras:

- **Figura 3-1: Diagrama de Arquitectura de ResNet50**
  https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758
- **Figura 3-2: Diagrama de Arquitectura de MobileNetV2**
  www.researchgate.net/figure/The-architecture-of-MobileNetV2-DNN_fig1_361260658
- **Figuras 3-3 y 3-4: Diagramas de Arquitectura de ConvNeXt-Tiny. Modificado de:**
  www.researchgate.net/figure/A-ConvNeXt-Tiny-overall-network-structure-B-ConvNeXt-Block-structure_fig2_367224906
- **Figura 3-5: Diagrama de Red Siamesa**
  www.researchgate.net/figure/Overview-of-the-Siamese-Neural-Network-architecture-using-triplet-loss-function_fig1_363780514
- **Figura 3-6: Diagrama de kNN**
  rajviishah.medium.com/introduction-to-k-nearest-neighbors-knn-algorithm-e8617a448fa8

## Enlaces importantes:

- **Github con los notebooks utilizados:**
  github.com/DiegoRoomero/CC_Thesis_DeepLearning
- **Documentación Keras Applications:**
  keras.io/api/applications
- **Red siamesa de ejemplos de Keras Models**
  keras.io/examples/vision/siamese_network/
- **Documentación librería skscikit-learnlearn, modelos kNN**
  sciki-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html