



Universidad de Castilla-La Mancha  
Escuela Superior de Ingeniería Informática

**Trabajo Fin de Grado**  
**Grado en Ingeniería Informática**  
Tecnología Específica de  
**Computación**

# Evaluación Estética de Fotografías: Modelos Basados en Aprendizaje Profundo y Transformers

Daniel Coronado Martín

Junio, 2022





**Trabajo Fin de Grado**  
**Grado en Ingeniería Informática**  
Tecnología Específica de  
**Computación**

# **Evaluación Estética de Fotografías: Modelos Basados en Aprendizaje Profundo y Transformers**

**Autor:** Daniel Coronado Martín  
**Tutor:** José Miguel Puerta Callejón  
**Co-Tutor:** Luis González Naharro

Junio, 2022

*Dedicado a mi familia y a toda la gente  
que hace las cosas con pasión*

## **Declaración de Autoría**

---

Yo, Daniel Coronado Martín con DNI 47400140S, declaro que soy el único autor del trabajo fin de grado titulado “Ingeniería Informática” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a 17 de junio de 2022

Fdo: Daniel Coronado Martín

## Resumen

---

Vivimos rodeados de contenido multimedia, tanto es así que las redes sociales más relevantes hoy en día son aquellas que basan la mayoría de su contenido en fotografías y vídeos: *Instagram, Tik Tok, Vinted o Tumblr*. Por ello, resolver problemas de Evaluación Estética de contenido multimedia como imágenes cada vez va ganando más importancia. Con algunas de sus principales aplicaciones como: la mejora de algoritmos de recomendación de publicaciones similares o la organización automática de fotografías almacenadas en la galería de nuestros teléfonos móviles.

Así pues, en este proyecto vamos a intentar resolver dicho problema a partir de técnicas conocidas del ámbito de la Visión Artificial como las Redes Convolucionales, además de modelos que lideran el Estado del Arte en este y otros problemas de Deep Learning, los Transformers.



## Índice general

---

<b>Capítulo 1</b>	<b>Introducción</b>	<b>1</b>
1.1	Motivación	1
1.2	Objetivos	1
1.3	Competencias previas	2
1.4	Estructura del proyecto	2
<b>Capítulo 2</b>	<b>Evolución de la Inteligencia Artificial</b>	<b>4</b>
2.1	Introducción	4
2.2	Modelos de Predicción	4
2.2.1	C4.5	5
2.2.2	Redes Neuronales	7
2.2.2.1	Hiper-parámetros	9
2.2.2.2	Visión Artificial	10
2.2.2.2.1	Redes Convolucionales	11
<b>Capítulo 3</b>	<b>Estado del Arte</b>	<b>15</b>
3.1	Introducción	15
3.2	Transformers	17
3.2.1	Procesamiento Natural del Lenguaje	17
3.2.2	Redes Neuronales Recurrentes	18
3.2.3	Mecanismos de auto-atención	19
3.2.4	Vision Transformers	21
3.3	ConvMixers	21
<b>Capítulo 4</b>	<b>Clasificación Estética de Imágenes</b>	<b>23</b>



4.1	Limitaciones del Problema	23
4.2	Metodología de trabajo	24
4.2.1	Bases de datos	24
4.2.1.1	CIFAR-10 y CIFAR-100	24
4.2.1.2	AVA	26
4.3	Metodología Agile SCRUM	26
4.3.1	1 <sup>er</sup> Hito: Experimentos previos sobre CIFAR-10	27
4.3.2	2 <sup>º</sup> Hito: Experimentos previos de Transfer Learning	30
4.3.3	3 <sup>er</sup> Hito: Experimentos y resultados finales sobre AVA	32
<b>Capítulo 5</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>35</b>
<b>Bibliografía</b>		<b>38</b>

## Índice de figuras

---

<b>Figura 1:</b>	Árbol C4.5 de ejemplo	5
<b>Figura 2:</b>	Esquema explicativo - Perceptrón artificial	7
<b>Figura 3:</b>	Función de activación logística	8
<b>Figura 4:</b>	Función de activación de umbral	8
<b>Figura 5:</b>	Función de activación ReLU	8
<b>Figura 6:</b>	Esquema explicativo - Perceptrón Multicapa	9
<b>Figura 7:</b>	Mal ajuste del learning rate	10
<b>Figura 8:</b>	Evolución de la Visión Artificial	11
<b>Figura 9:</b>	Matriz intensidad de píxeles para Escala de Grises	12
<b>Figura 10:</b>	Matrices intensidad de píxeles para RGB	12
<b>Figura 11:</b>	Esquema explicativo - Flujo red convolucional	13
<b>Figura 12:</b>	Evolución de características convolucionales - 1	14
<b>Figura 13:</b>	Evolución de características convolucionales - 2	14
<b>Figura 14:</b>	Gráfica – Benchmark ImageNet	16

<b>Figura 15:</b> Gráfica – Benchmark AQA	17
<b>Figura 16:</b> Esquema explicativo - Redes Neuronales Recurrentes	18
<b>Figura 17:</b> Doble análisis profundo de la palabra "pangolín"	19
<b>Figura 18:</b> Vector Atención de "El" dentro de "El pangolín duerme en su árbol"	19
<b>Figura 19:</b> Matriz Atención resultante de "El pangolín duerme en su árbol"	19
<b>Figura 20:</b> Flujo profundo - Cálculo y ajuste del Vector Output	20
<b>Figura 21:</b> Esquema explicativo - Vision Transformers	21
<b>Figura 22:</b> Esquema Explicativo - ConvMixers	21
<b>Figura 23:</b> SCRUM board	27
<b>Figura 24:</b> Hito 1 - gestor de proyectos GitHub	27
<b>Figura 25:</b> Evolución del modelo ViT sobre CIFAR-10	28
<b>Figura 26:</b> Demostración de la precisión del ConvMixer en CIFAR-10	29
<b>Figura 27:</b> Evolución del modelo ConvMixer sobre CIFAR-10	29
<b>Figura 28:</b> Hito 2 - gestor de proyectos GitHub	30
<b>Figura 29:</b> Evolución del modelo ViT sobre CIFAR-100	30
<b>Figura 30:</b> Evolución del modelo ConvMixer sobre CIFAR-100	31
<b>Figura 31:</b> Hito 3 - gestor de proyectos GitHub	32
<b>Figura 32:</b> Evolución del ECM en ViT sobre AVA	32
<b>Figura 33:</b> Evolución del ECM en ViT sobre AVA - Validación	33
<b>Figura 34:</b> Evolución del ECM en ConvMixer sobre AVA	34

## Índice de tablas

---

<b>Tabla 1:</b> Etiquetas CIFAR-10	24
<b>Tabla 2:</b> Etiquetas CIFAR-100	25



# Capítulo 1

## Introducción

---

### 1.1 Motivación

La Inteligencia Artificial avanza día a día resolviendo cada vez problemas más y más complejos.

En estos últimos años hemos visto algoritmos capaces de detectar la existencia de tumores en radiografías ([Gokila Brindha et al., 2021](#)), entender y redactar textos como lo haría un humano ([Iqbal & Qureshi, 2020](#)), crear imágenes hiperrealistas ([Ramesh, 2022](#)),... todo ello con una precisión asombrosa.

En mi caso, vamos a tratar un problema de Visión Artificial de carácter subjetivo. Vamos a intentar conseguir un modelo de Red Profunda capaz de clasificar imágenes según si son “bonitas” o “feas”.

Esto lo haremos a través del acercamiento a uno de los algoritmos considerados Estado del Arte en la actualidad, los Transformers; y aquellos que basan su potencia en el análisis automático de características visuales a través de capas convolucionales, los ConvMixers.

### 1.2 Objetivos

Por tanto, en la división de objetivos para alcanzar este reto podríamos distinguir:

1. Hacer computacional la subjetividad de una opinión personal sobre la calidad estética de una imagen.
2. Implementar un modelo suficientemente preciso que resuelva nuestro problema

El primero de ellos lo conseguiremos con la solución que proponen en AVA: recopilar los votos recibidos en concursos de fotografía sobre las instantáneas que formarán la base de datos con la que trabajaremos.

Y el segundo, a través de técnicas de Redes Profundas y Convolucionales, junto con mecanismos de auto-atención que formarán nuestros modelos finales ConvMixer y ViT, respectivamente.

### 1.3 Competencias previas

Para lograr los objetivos explicados en el punto anterior requeriremos de algunas competencias y habilidades previas que hemos obtenido durante el grado:

- Uso de herramientas de gestión de desarrollo como GitHub. Usaremos un repositorio GitHub ya no solo para almacenar las implementaciones y resultados que vayamos obteniendo, sino para organizar temporalmente el desarrollo del proyecto, también.
- Conocimiento previo en Python. Por ser el lenguaje de programación sobre el que realizaremos todas las implementaciones. Junto con las distintas librerías de machine learning e indexación de datos que conocemos.
- Conocimiento sobre construcción de contenedores e imágenes Docker. Por su importancia para lanzar experimentos en entornos locales y cluster sin necesidad de afectar al resto de usuarios de los mismos. Además de poder cumplir con la reproducibilidad de los experimentos.
- Capacidad de investigación y recopilación de información. Por su contemporánea y veloz evolución, necesitamos estar al día de los últimos desarrollos sobre Deep Learning y Visión Artificial.

Además de algunas otras como dominio del lenguaje Bash Script de Linux o conocimiento previo de procesamiento digital de imágenes.

### 1.4 Estructura del proyecto

Primero, en todo el [apartado 2](#), haremos un paseo a través de la evolución algorítmica de la Inteligencia Artificial a lo largo de este último siglo. Partiendo de clasificadores básicos como el C4.5, visto en las asignaturas de Sistemas Inteligentes y Sistemas Basados en el Conocimiento, y llegando hasta aquellos que suponen el Estado del Arte en la actualidad ([apartado 3](#)), los Transformers y ConvMixers. Además de explicar, por su potencia e importancia en la escena actual de la Inteligencia Artificial, las Redes Neuronales y Convolucionales.

Todo esto con el objetivo de introducir el problema de la clasificación subjetiva de imágenes, sus particularidades y las dificultades que, por su carácter, supone ([apartado 4](#)). Justificando, por tanto, las propuestas que presentamos en este proyecto: utilizar modelos pre-entrenados en problemas complejos de clasificación de imágenes. Tanto con aquellos que siguen la estructura puntera de los Transformers de Visión, como los que basan su potencia en las capas de convolución simplemente.

Durante la evolución del proyecto usaremos bases de datos de clasificación de imágenes simples como CIFAR-10 y CIFAR-100 como benchmark de la precisión de los modelos básicos y pre-entrenados ([apartado 4.3.1](#) y [4.3.2](#)). Así, en una metodología evolutiva SCRUM, iremos progresando y validando la eficacia de los modelos hasta ser capaces de clasificar sobre el conjunto objetivo, AVA (Aesthetic Visual Assessment) ([apartado 4.3.3](#)). Toda esta progresión se irá haciendo de forma paralela al análisis de los resultados de cada experimento en cada hito.

# Capítulo 2

## Evolución de la Inteligencia Artificial

---

### 2.1 Introducción

Plantearse siquiera un problema así era impensable hace un siglo, por lo que me parece de especial mención nombrar la evolución algorítmica que hemos ido viviendo.

Explicando paso a paso: desde los clasificadores más básicos que resuelven con suficiente precisión problemas de predicción; hasta los algoritmos contemporáneos considerados Estado del Arte, pasando por las Redes Neuronales y Convolucionales.

### 2.2 Modelos de Predicción

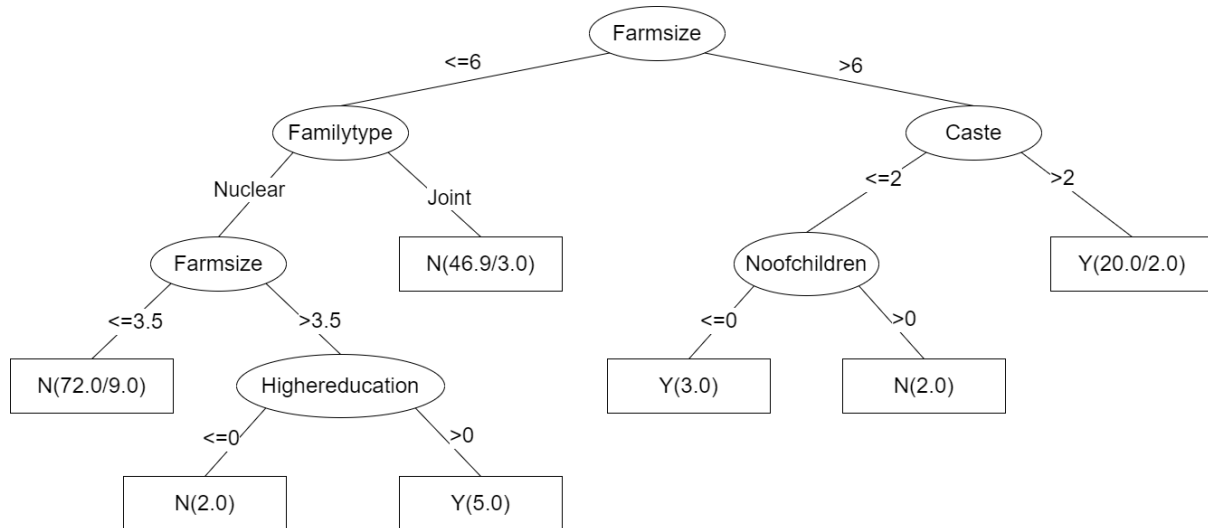
Un modelo de predicción es aquel algoritmo que previo entreno es capaz de pronosticar el resultado de un problema futuro. Aquí entrarían, por ejemplo, los problemas de predicción meteorológica, mercado en bolsa, resultados deportivos, etc.

Todos ellos se basan en, a partir de una base de datos de entreno, generar reglas/patrones con los que posteriormente clasificar un caso de entrada nuevo. Es decir, por ejemplo en el caso de la predicción meteorológica, la base de datos tiene diversas características como pueden ser la velocidad y dirección del viento, la humedad, la presión atmosférica, etc. de un día concreto. Normalmente en la misma base de datos encontramos la variable clase, que es por la que pretendemos predecir. Es decir, si llovió ese día o no. Por tanto, cuando nuestro algoritmo haya aprendido esos patrones de los que hablábamos antes, sabrá que, dadas unas características meteorológicas particulares, lloverá o no.



Este podría ser por ejemplo el caso del algoritmo C4.5 ([Quinlan, 1992](#)).

### 2.2.1 C4.5



**Figura 1:** Árbol C4.5 de ejemplo

Que primero busca, a través del cálculo de la ganancia de información, cuál es la característica que más conocimiento nos proporciona, la selecciona y siguiendo su dominio repite el paso anterior con las características restantes.

En el caso de tratarse de un problema con solamente variables categóricas, es decir con un dominio finito, las ramificaciones del árbol C4.5 serían tantas como elementos en el dominio de cada variable. Sin embargo, en el caso de problemas con variables numéricas también, como es el que mostramos en la **Figura 1** debemos elegir cual es el rango numérico que más ganancia de información nos aporta, pudiendo, como es lógico, repetir más de una vez la misma característica, pero con un rango distinto.

Así, vamos iterando el algoritmo hasta que obtenemos un conjunto que dadas unas características es completamente de una clase (Y o N), es decir, en nuestro ejemplo de la predicción meteorológica un conjunto de registros de la base de datos en los que llovería o no. A este conjunto se le conoce como hoja del árbol de clasificación.

Por tanto, el árbol de clasificación generado durante la iteración del algoritmo C4.5 puede ser traducido como Sistema Basado en Reglas, en el que cada camino de ramas que lleva hasta una hoja supone una regla.

En el caso de la **Figura 1**, la regla que correspondería con la hoja de más a la izquierda sería como prosigue:

*Farmsize ( $\leq 6$ ) AND Familytype (Nuclear) AND Farmsize ( $\leq 3.5$ ) THEN N*

Como podemos observar, en este caso la hoja contiene registros de ambas clases, pero al contener un mayor número de registros clasificados como N que de aquellos clasificados como Y la predicción es N.

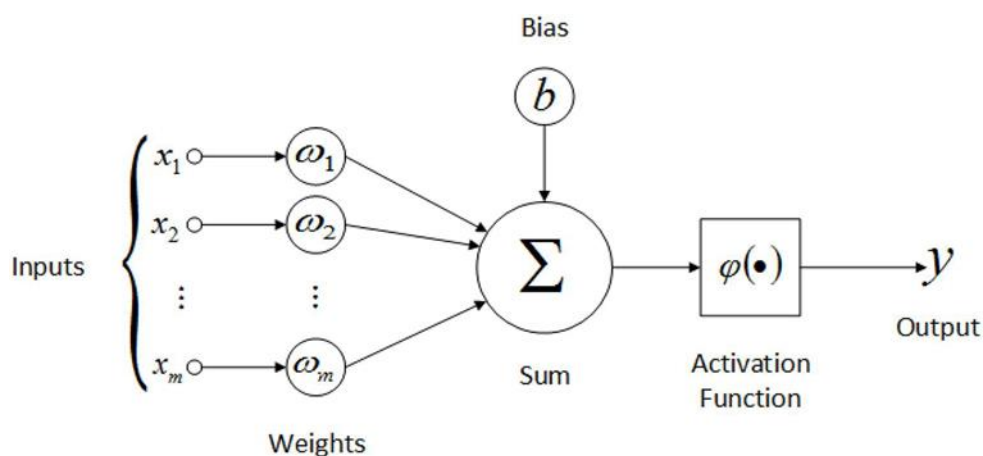
Existen problemas con este tipo de algoritmos de clasificación debido a que podemos encontrar casos en los que el conjunto de registros clasificados en una hoja sea demasiado pequeño y concreto. Imaginemos un despliegue de este árbol mucho mayor con hojas que contuvieran un único registro. Eso significaría que estaríamos haciendo una regla por cada registro de nuestra base de datos de entrenamiento lo que supone un gran coste computacional y un alto sobreajuste. Y el problema del sobreajuste no es algo trivial en el mundo del Machine Learning. Debemos evitar ante todo sobreajustar a la base de datos de entrenamiento y hacer a nuestro algoritmo general, capaz de recibir cualquier caso particular nuevo y poder procesarlo y predecir correctamente a partir de él.

Para ello, hacemos uso de, entre otras técnicas e hiper-parámetros, aquellas que detienen nuestra ejecución antes de llegar a un caso extremadamente concreto del conjunto de registros por hoja.

## 2.2.2 Redes Neuronales

Del mismo carácter clasificativo que el algoritmo C4.5 son, por ejemplo, las tan conocidas Redes Neuronales ([Hopfield, 1982](#)).

En este caso se trata de una estructura formada por perceptrones o neuronas artificiales.



**Figura 2:** Esquema explicativo - Perceptrón artificial

Un perceptrón (**Figura 2**) recibe parámetros de entrada con un peso, aplica en su interior la suma ponderada de estos y devuelve una salida; resultando en una función lineal básica. Esta suma ponderada tiene la misma estructura que una regresión lineal:

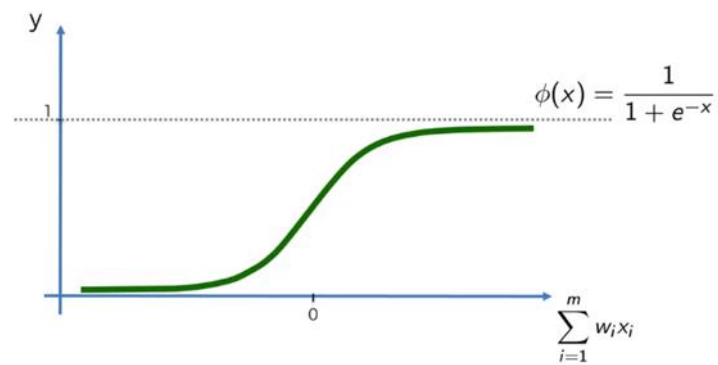
$$Sum = X1*W1 + X2*W2 + ... + Xm*Wm + Bias$$

Donde cada una de las variables explicativas o regresoras son las variables de entrada:  $X_1, X_2, \dots, X_m$ ; los parámetros del modelo de la regresión lineal, los pesos por los que multiplicamos cada variable:  $W_1, W_2, \dots, W_m$ ; y el Bias o Sesgo el término constante o intersección de la regresión lineal.

A esa función lineal resultante se le aplica entonces una función de activación. Esta se encargará de darle forma a los resultados de la regresión lineal y producir la salida final de la neurona.

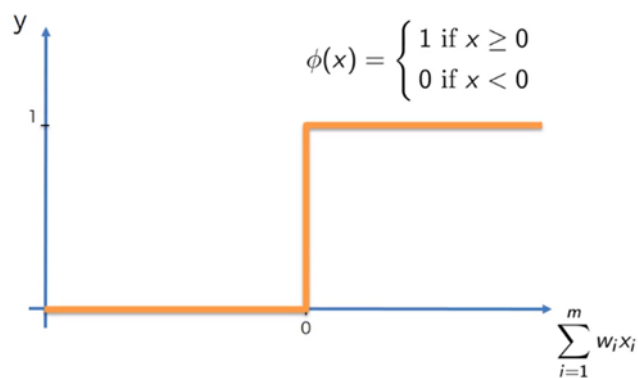
Conocemos de entre otras, las siguientes funciones de activación:

- Logística o sigmoide



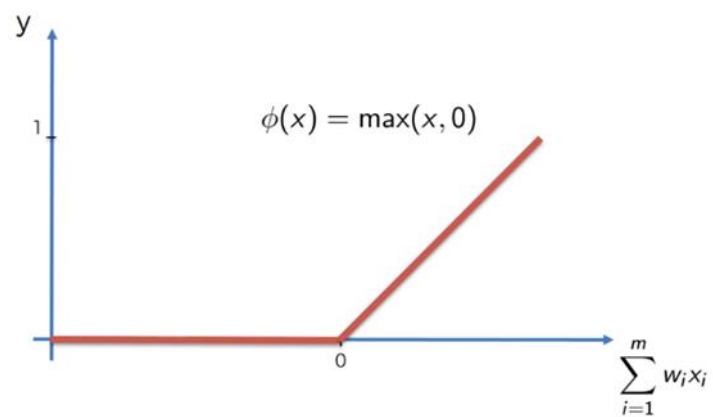
**Figura 3:** Función de activación logística

- Threshold (umbral) o Step Function



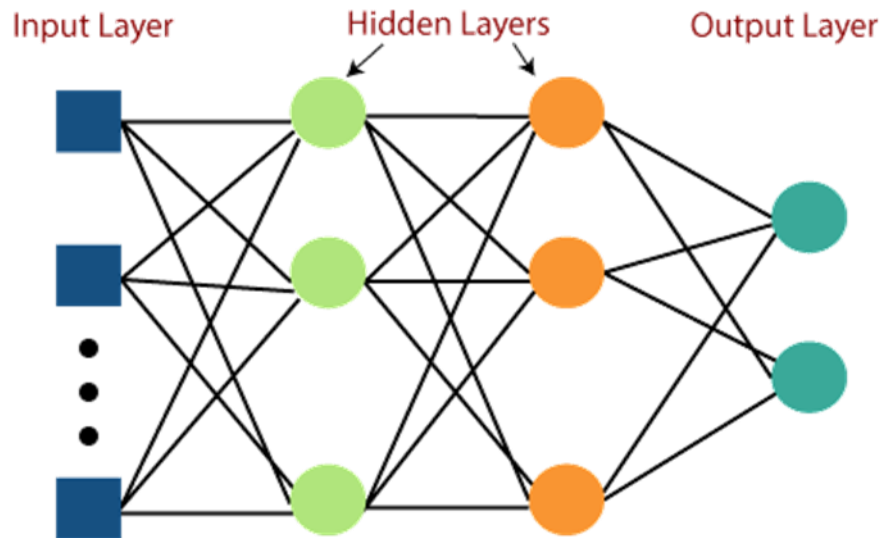
**Figura 4:** Función de activación de umbral

- Rectified Linear Unit o ReLU



**Figura 5:** Función de activación ReLU

Finalmente, la salida individual de la neurona será enviada a otra capa de la red neuronal como variable de entrada de otra neurona que repetirá el mismo proceso, o bien responderá como la salida final de nuestro sistema (**Figura 6**).



**Figura 6:** Esquema explicativo - Perceptrón Multicapa

A través de la retropropagación, la red neuronal comprobará si el resultado final del sistema se corresponde con la salida esperada. En caso contrario, propagará hacia atrás el ajuste de los pesos individuales de cada variable de entrada. Esto lo hará por cada neurona de cada capa de nuestro sistema.

Reiterando este proceso durante la fase de entreno, obtenemos un modelo final completamente ajustado a las salidas deseadas.

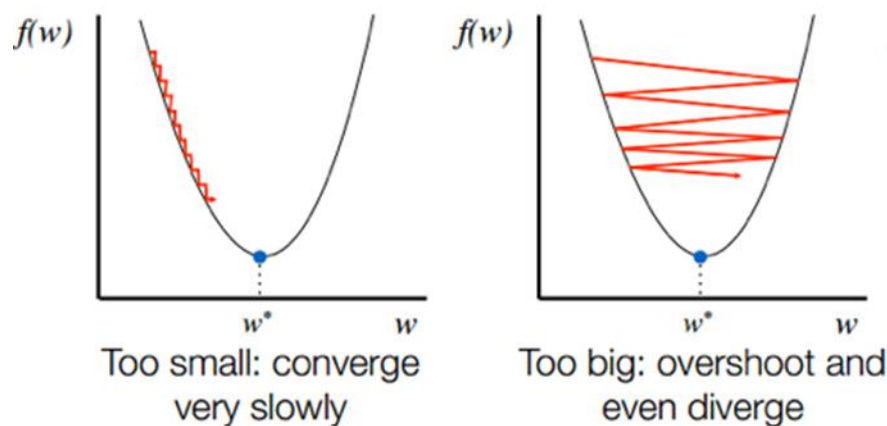
### 2.2.2.1 Hiper-parámetros

Como decíamos en el apartado anterior los clasificadores suelen hacer uso de distintos hiper-parámetros en su entreno ([Yang & Shami, 2020](#)). Variables que ayudarán al desarrollador a ajustar el modelo según las características propias e intuición tras su estructura.

En el caso del Deep Learning, hablamos por ejemplo de:

- **Learning rate** (o tasa de aprendizaje), que fija cuánto van a variar los pesos internos de la red a lo largo de su entreno.

Así, la evolución del gradiente descendiente, que busca ajustar los pesos del modelo al dominio que definen las etiquetas de nuestro problema, se hace progresivamente.



**Figura 7:** Mal ajuste del learning rate

En el caso de escoger un learning rate demasiado bajo, la red convergerá muy lentamente. Análogamente, al fijarlo demasiado alto tenderá a excederse e incluso divergir de los resultados esperados.

- **Number of epochs** (o número de iteraciones), establece la cantidad de iteraciones que queremos que nuestra red haga sobre el conjunto total de entrenamiento.
- **Batch size** (o tamaño de batch), divide el conjunto de entrenamiento en carpetas o “batches” sobre las que iterará durante una misma epoch.

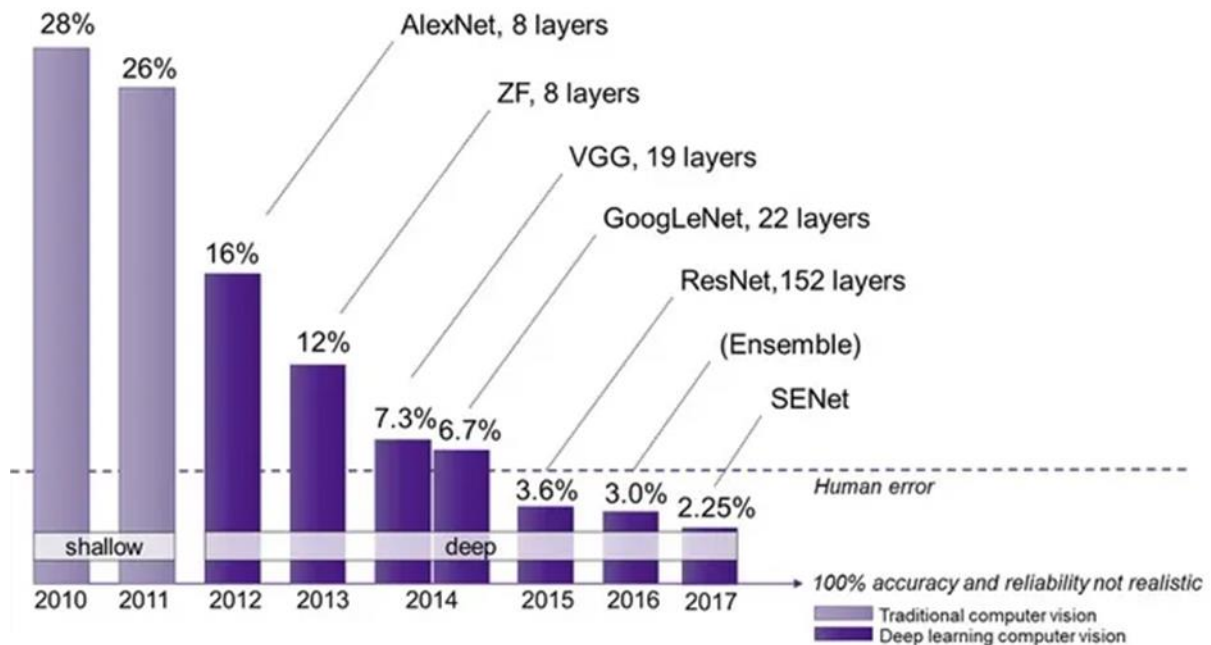
#### 2.2.2.2 Visión Artificial

Este subapartado figura como parte de 2.2.2 *Redes Neuronales*, no por cuestión obvia. De hecho, si echamos la vista atrás a otros TFG de ex-compañeros míos como [Chamón \(2014\)](#), la Visión Artificial era tratada como un ámbito completamente independiente a las redes profundas.

La Visión Artificial era utilizada en su proyecto para la extracción manual de características semánticas y gráficas de la imagen. Usando técnicas de procesamiento digital de imágenes buscaba hacer computacional el enfoque, el rango de colores o la nitidez.

Estas características eran las que posteriormente formaban el conjunto de datos sobre el que trabajaría el clasificador. Por tanto, como podemos ver, el flujo de trabajo de la Visión Artificial no era completamente automático. Dependía fuertemente del científico de datos y de la calidad del conocimiento que lograra extraer. Es por esto que muchas veces formaban equipo con especialistas del ámbito del problema en cuestión.

No sería hasta después de 2015, con los primeros resultados verdaderamente sorprendentes de Visión Artificial a través de redes profundas, que el rol del científico de datos caería a un segundo lugar ([Cooper, 2019](#)).



**Figura 8:** Evolución de la Visión Artificial

A partir de entonces, todo el análisis y detección de características visuales se integraría directamente sobre el aprendizaje profundo y automático de las redes neuronales, dando lugar a las que conocemos hoy en día como Redes Convolucionales.

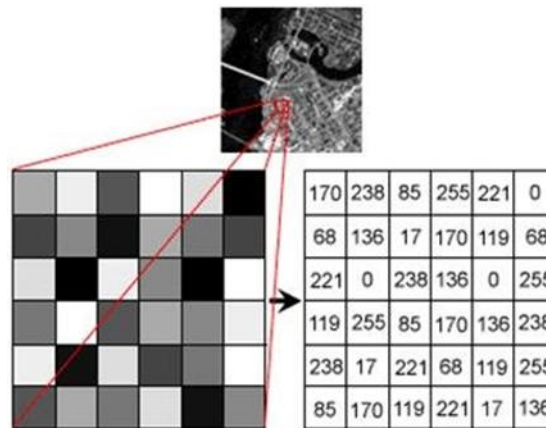
#### 2.2.2.2.1 Redes Convolucionales

Al igual que cuando hablábamos de problemas de redes neuronales nuestro objetivo era extraer los grados de dependencia existentes entre las variables de entrada y los resultados esperados, en los del ámbito de la visión artificial será obtener las características visuales que identifican a una imagen con una etiqueta.

Para ello, necesitaremos representar las imágenes en estructuras matemáticas sobre las que nuestros sistemas puedan indexar y trabajar. Haciendo uso, por tanto, de matrices de igual tamaño que las dimensiones en píxeles de la imagen. Y almacenando en ellas los valores individuales de intensidad lumínica de cada pixel.

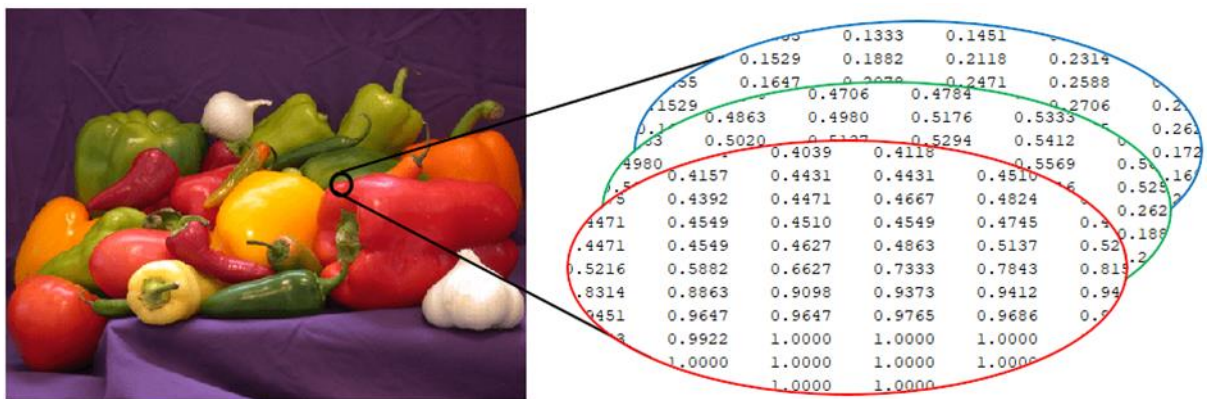
Esto lo haremos tantas veces como canales de entrada tenga la imagen:

- 1 canal - Escala de grises:



**Figura 9:** Matriz intensidad de píxeles para Escala de Grises

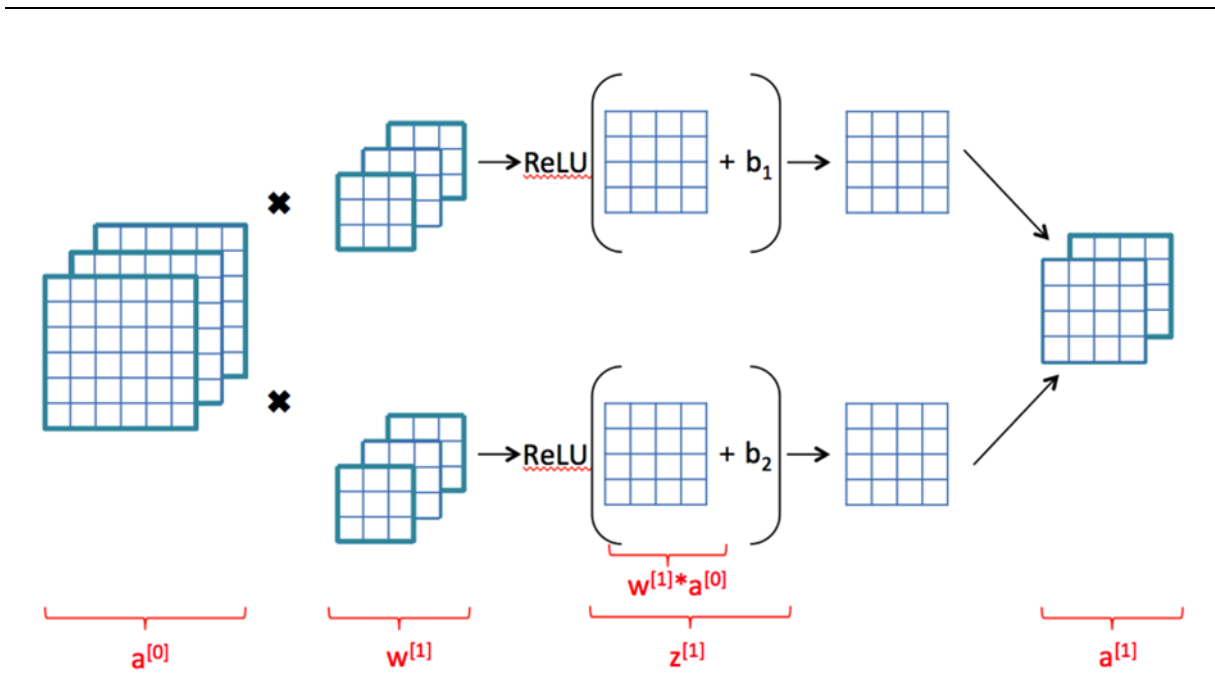
- 3 canales - RGB:



**Figura 10:** Matrices intensidad de píxeles para RGB

A estas matrices se les aplicarán iterativamente filtros convolucionales que actuarán como pesos a ajustar en la retropropagación de la red.



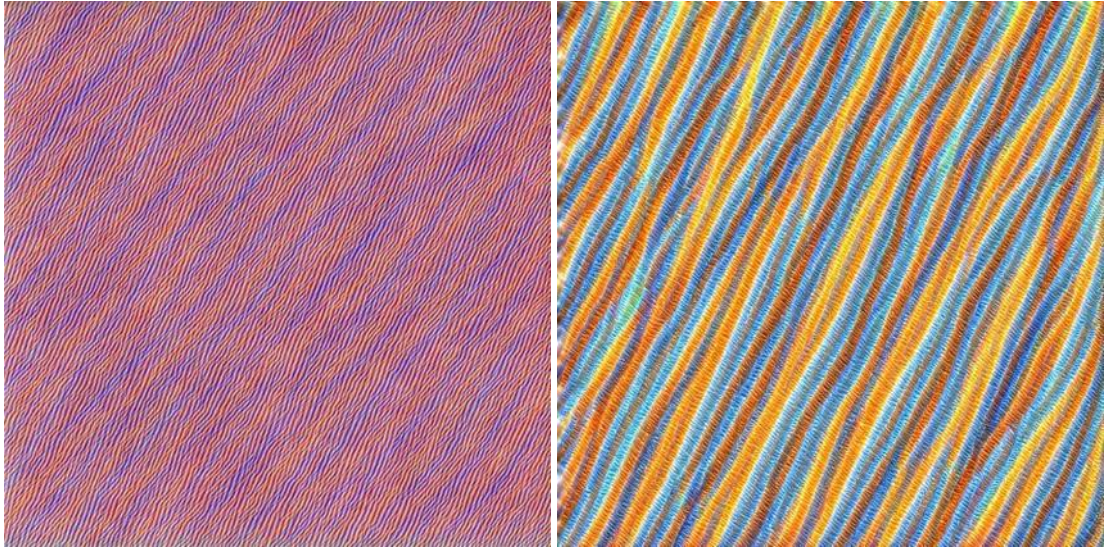


**Figura 11:** Esquema explicativo - Flujo red convolucional

Por tanto, las matrices que produzcan los filtros convolucionales ( $w*a$ ) en cada canal de entrada de la imagen serán sumadas entre sí y con un sesgo ( $b$ ). Produciendo, al paso por la función de activación, la salida final de la neurona (**Figura 11**) ([Cavaioni, 2018](#)).

Como podemos observar, hemos conseguido adaptar totalmente el flujo de trabajo de la red neuronal a las peculiaridades de utilizar como entrada imágenes en lugar de parámetros o variables numéricas.

De forma anecdótica y por la estructura matricial de pesos ajustables que suponen los filtros convolucionales, podemos visualizar la composición gráfica que va haciendo la red a lo largo de sus capas ([Graetz, 2021](#)). Fijándose en las primeras en elementos adyacentes del espacio y construyendo líneas o trazos simples:



**Figura 12:** Evolución de características convolucionales - 1

Generando figuras más y más complejas conforme profundizamos en la red:



**Figura 13:** Evolución de características convolucionales - 2

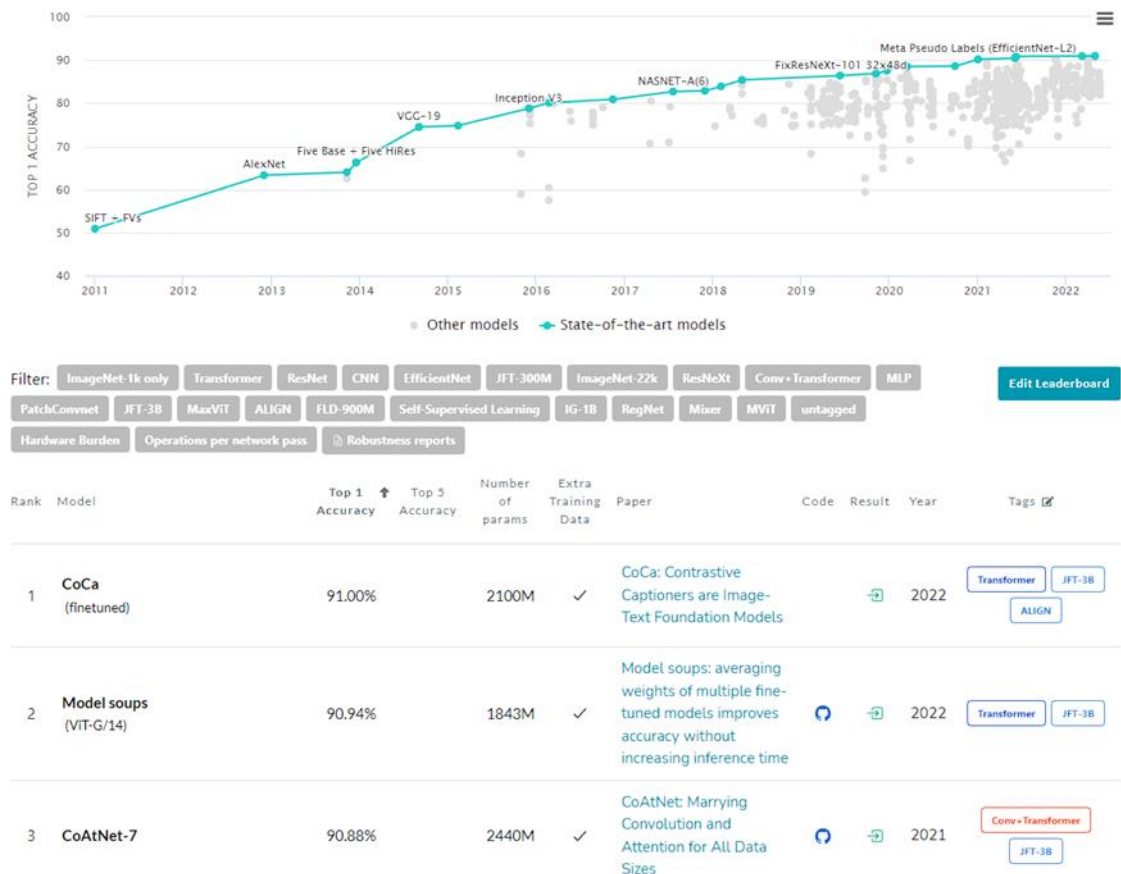
# Capítulo 3

## Estado del Arte

---

### 3.1 Introducción

No podemos hablar del Estado del Arte en Clasificación de Imágenes sin mencionar ImageNet. Esta base de datos majestuosa contiene más de 14 millones de imágenes etiquetadas de entre 20 mil categorías. Como podemos imaginar, supone uno de los mayores retos para la Visión Artificial en la actualidad. Por lo que miles de investigadores intentan día a día conseguir resultados cada vez mejores en su clasificación automática.

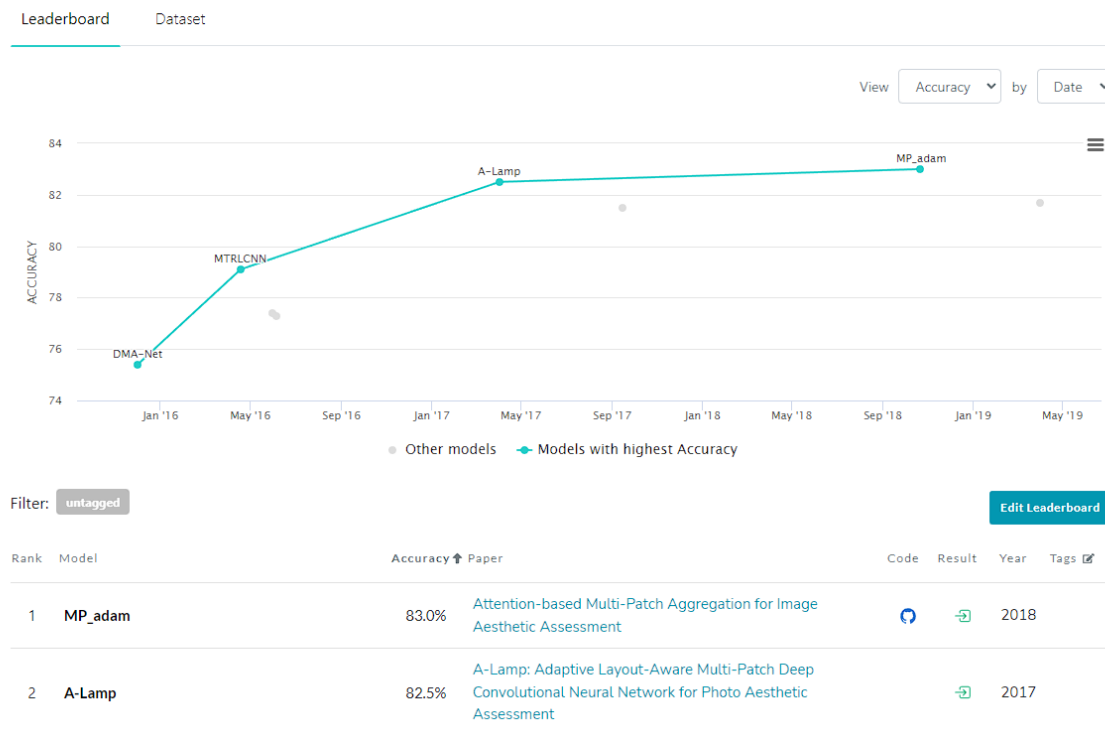


**Figura 14:** Gráfica – Benchmark ImageNet

Los modelos más destacados en este problema son, por ejemplo: ViT-G/14, un modelo Vision Transformer entrenado por Google ([Zhai, 2021](#)); o CoAtNet, una mezcla de redes convolucionales con capas de auto-Atención ([Dai, 2021](#)). Ambos dos están basados en las estructuras que vamos a tratar durante el desarrollo de este proyecto.

Por otro lado, la evolución en el carácter estético y subjetivo de la clasificación de imágenes ha ido principalmente marcada por AVA. La base de datos “Aesthetic Visual Assessment” o Evaluación Estética Visual. En la que según la competición “Aesthetics Quality Assessment on AVA” podemos comprobar, otra vez, la importancia de las redes profundas convolucionales y el uso de capas y mecanismos de atención. Es el caso por ejemplo de A-Lamp y MP-adam (**Figura 15**).

## Aesthetics Quality Assessment on AVA



**Figura 15:** Gráfica – Benchmark AQA

## 3.2 Transformers

Como hemos dicho, los progresos más relevantes en la Clasificación de Imágenes se obtienen a través de mecanismos de atención. En ellos están basados los Transformers, que consiguen resultados muy punteros en problemas de Visión Artificial (ViT-G) pero sobre todo en Procesamiento Natural del Lenguaje (GPT-3). Por ello y por su relevancia en el desarrollo de este proyecto me parece conveniente detenernos y revisar su estructura.

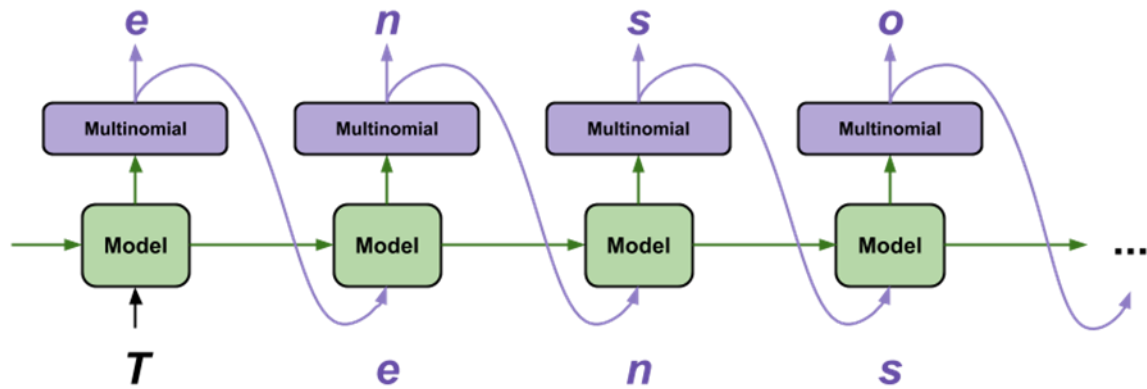
### 3.2.1 Procesamiento Natural del Lenguaje

La motivación tras los mecanismos de auto-atención surge de los problemas y limitaciones que se conocían hasta la fecha en el ámbito del Procesamiento Natural del Lenguaje.

Este ámbito del machine learning busca automatizar la comprensión y redacción de textos complejos a través del aprendizaje profundo. Con ello se obtienen grandes avances en aplicaciones de traducción precisa, generación automática de textos, buscadores web, filtros de correo electrónico...

### 3.2.2 Redes Neuronales Recurrentes

Anteriormente, los problemas NLP (Natural Language Processing) se abordaban a través de Redes Neuronales Recurrentes (**Figura 16**).



**Figura 16:** Esquema explicativo - Redes Neuronales Recurrentes

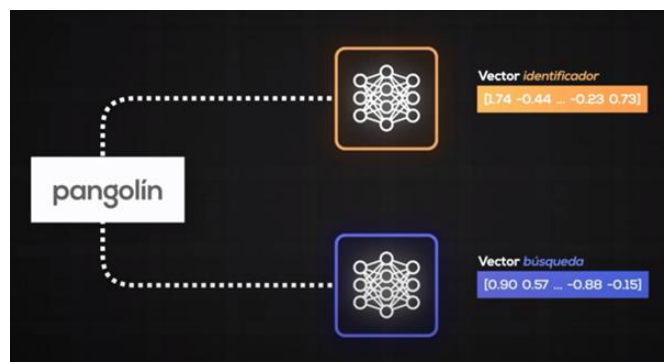
Estas basan su estructura en analizar secuencialmente cada una de las palabras del texto objetivo a través de un modelo de red profunda. La salida del modelo de la primera palabra alimenta la entrada del de la segunda y así sucesivamente; de forma similar a como cualquier persona lee un texto palabra por palabra. Así, mantenemos la dependencia ordinal de las palabras durante su análisis y procesamiento computacional.

Sin embargo, esta estructura presenta una gran limitación: al paso reiterado de palabras por la Red Recurrente, el peso de las más antiguas va siendo cada vez menos significativo, lo que hace que en frases como: “El pangolín dormía plácidamente colgado de la rama de un árbol usando su cola”, la dependencia de “El pangolín” con “su cola” se pierda completamente a lo largo de su análisis. Por ello nacen los mecanismos de auto-atención ([Santana, 2021](#)).



### 3.2.3 Mecanismos de auto-atención

A partir de ahora, cada palabra será analizada por dos redes neuronales: una que nos devolverá el Vector Identificador (o key) con las propiedades características de la palabra; y otra, el Vector Búsqueda (o query) con las propiedades de interés para dicha palabra (**Figura 17**).



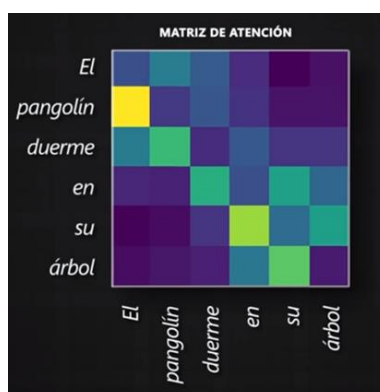
**Figura 17:** Doble análisis profundo de la palabra "pangolín"

Para analizar la dependencia de una palabra con las del resto de su frase, solamente nos queda calcular el Producto Escalar de su vector "query" por todos los "key" de las demás.



**Figura 18:** Vector Atención de "El" dentro de "El pangolín duerme en su árbol"

Obteniendo así el Vector de Atención de cada una de nuestras palabras y, por ende, la Matriz Atención del modelo.



**Figura 19:** Matriz Atención resultante de "El pangolín duerme en su árbol"

Una vez tengamos la Matriz de Atención solo falta seguir el flujo típico del aprendizaje profundo, es decir, entrenar sobre el problema objetivo. En este caso, la red neuronal, devuelve por cada palabra un Vector Valor que será sumado ponderadamente con el del resto de las palabras según su Vector de Atención:



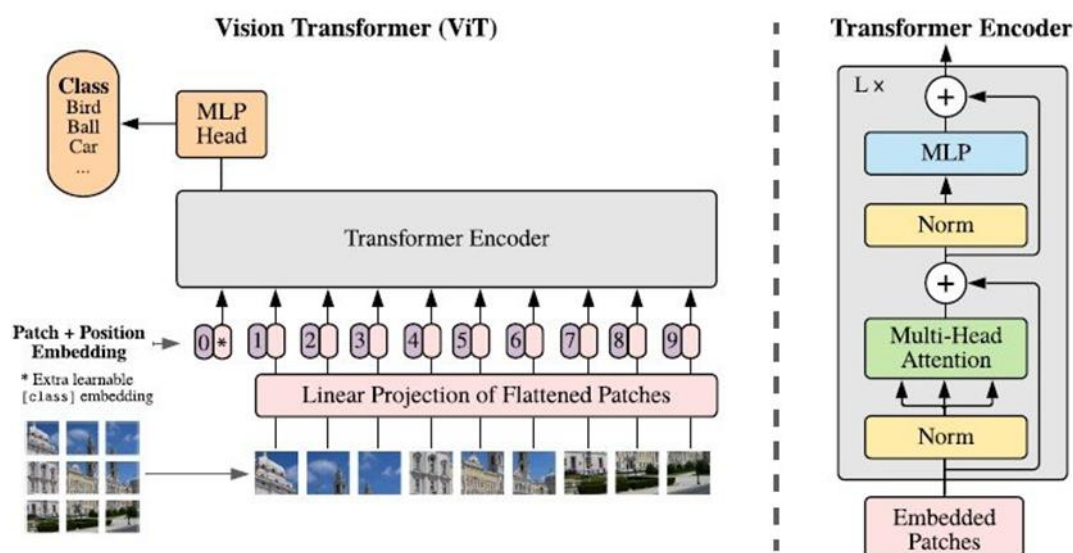
**Figura 20:** Flujo profundo - Cálculo y ajuste del Vector Output

Obteniendo, finalmente, el Vector Output sobre el que se hará el proceso de retropropagación que comentábamos en el apartado 2.2.2. Así ajustaríamos los pesos internos de esta última red neuronal y nuestro modelo aprendería ([Santana, 2021](#)).



### 3.2.4 Vision Transformers

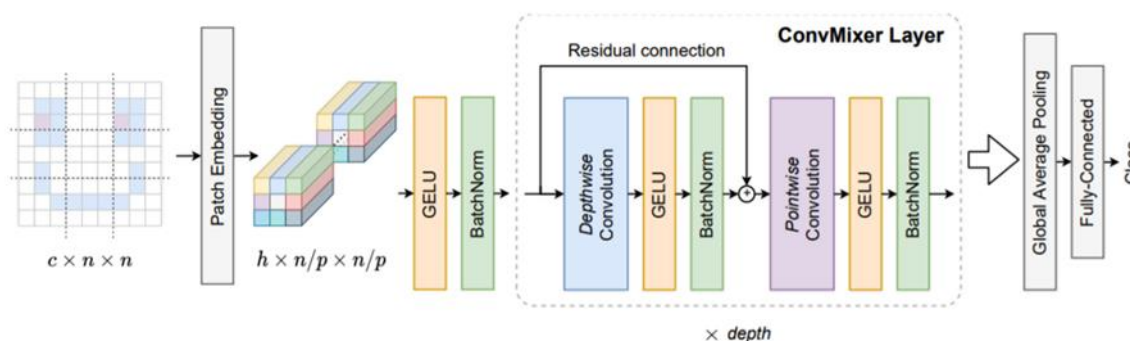
Esta misma metodología se aplica en los Transformers de problemas de Visión Artificial. En los que, en lugar de tener palabras como elementos de entrada, tenemos píxeles; y en lugar de estructuras secuenciales de palabras dentro de un texto, tenemos estructuras espaciales de parches de una imagen.



**Figura 21:** Esquema explicativo - Vision Transformers

En este caso, y por la característica potencia de cómputo en paralelo que tienen los Transformers, necesitamos conservar el orden espacial de los parches que se introducirán simultáneamente sobre el input del modelo (Patch + Position Embedding) ([Dosovitskiy, 2020](#)).

### 3.3 ConvMixers



**Figura 22:** Esquema Explicativo - ConvMixers

Como podemos observar, una vez más, dividimos la imagen de entrada en parches a través de la capa de Patch Embedding. A continuación, aplicamos una red profunda de capas convolucionales que se irán alternando entre “depth-wise” y “point-wise”.

Las primeras se encargarán de aplicar filtros convolucionales a cada canal de la imagen por separado. A diferencia de las capas convolucionales clásicas que aplican el mismo filtro a todos los canales por igual. Seguida a estas encontramos las “point-wise”, que aplican filtros de tamaño 1x1 pixel. Esta estructura se repite reiteradas veces a lo largo de la creación primigenia del modelo.

La intuición tras el uso de este tipo de estructuras es focalizar sobre la detección de características en la capa “depth-wise” y combinar linealmente los resultados de los distintos canales a través de las “point-wise” ([Park, 2022](#)).

Por último y como veíamos en las Redes Recurrentes, al generar un modelo muy profundo en número de capas, tenemos que prestar especial atención a posibles problemas como el desvanecimiento del gradiente en su retropropagación. En este caso, solucionamos el problema a través de una conexión residual entre entradas de distintas capas. Así, conservamos el peso de las salidas de capas anteriores a lo largo de su secuencialidad.

# Capítulo 4

## Clasificación Estética de Imágenes

---

### 4.1 Limitaciones del Problema

Durante mi paso por las asignaturas del ámbito de la Inteligencia Artificial y el Machine Learning, los problemas que se nos planteaban a mis compañeros y a mí eran de carácter objetivo: predecir la lluvia en Australia o hacer un estudio de las variables que determinarían si los tripulantes del Titanic sobrevivirían o no. Ambos dos cuentan con una base de datos de valores concretos que nos hacen deducir reglas o patrones determinantes para su resultado. También ocurría lo mismo en aquellos de Visión Artificial, clasificar imágenes según si en ellas aparecían coches, pájaros, ciervos... (CIFAR-10) o detectar un número del 0 al 9 escrito a mano (MNIST).

En el caso de AVA (Aesthetic Visual Assessment), el problema cambia completamente de paradigma; tratando de etiquetar imágenes de forma completamente subjetiva. Utilizando, para ello, un vector de nueve posiciones por instantánea en el que cada posición representa una nota del 1 al 10. Cada imagen proviene de un reto de fotografía profesional y amateur distinto. Tras su finalización, se obtienen los vectores con todos los votos de los usuarios según si la instantánea ha gustado más o menos. Así obtenemos una etiqueta que representa la calidad estética subjetiva de cada una de nuestras fotografías.

Sin embargo, como podemos imaginar, no es suficiente un vector de nueve posiciones para dotar a una Inteligencia Artificial de criterio y opinión sobre si una foto le gusta o no. La subjetividad es una cualidad humana que no va ligada a ningún factor o variable objetivizable. Por ello, los resultados finales que obtengamos deberán ser vistos desde lo abstracto de una red que busca ajustar su conocimiento al dominio del problema y ligarlo con las características visuales que aprenda.

## 4.2 Metodología de trabajo





































































































Antes de pasar a explicar la metodología seguida durante el desarrollo del proyecto que defiende este trabajo, me gustaría que nos detuviéramos un segundo en analizar la estructura de las bases de datos que escogemos para el mismo, sus especificaciones y el motivo tras su uso.

### 4.2.1 Bases de datos

#### 4.2.1.1 CIFAR-10 y CIFAR-100

Ambas creadas por el Canadian Institute for Advanced Research, de ahí su nombre, contienen 60.000 imágenes RGB de tamaño 32x32.

En el caso de CIFAR-10, cada una de sus 10 clases representa:

<b>avión</b>										
<b>automóvil</b>										
<b>pájaro</b>										
<b>gato</b>										
<b>ciervo</b>										
<b>perro</b>										
<b>rana</b>										
<b>caballo</b>										
<b>barco</b>										
<b>camión</b>										

**Tabla 1:** Etiquetas CIFAR-10

Y en el caso de CIFAR-100:

<b>Superclass</b>	<b>Classes</b>
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

**Tabla 2:** Etiquetas CIFAR-100

Como vemos, las 60.000 imágenes de CIFAR-100 pueden ser divididas en 20 superclases, generando una base de datos ligeramente más compleja que su hermana pequeña, CIFAR-10. En la que los subconjuntos de train y test pasan de contener 500 a 250 imágenes por etiqueta y 100 a 50 imágenes por etiqueta, respectivamente.

O bien, en 100 divisiones, quedando solamente 50 imágenes por clase en la parte de entreno y 10 en la parte de test. Cuando nos encontramos con bases de datos demasiado pequeñas en número de registros por etiqueta se suele dar graves problemas como: sobreajuste, alto ruido o alta complejidad en los modelos resultantes. A este problema se le conoce como Lack of data o falta de datos y supone un handicap a tener en cuenta para nuestros modelos y sus resultados.

Por ello, escogemos CIFAR-10 como base de datos de partida que nos mostrará si los modelos básicos son prometedores; y CIFAR-100 como indicadora de la evolución de los modelos a lo largo del proyecto.

#### 4.2.1.2 AVA

Como ya comentábamos en el apartado 3.1, AVA figura como la principal base de datos para clasificación estética de imágenes.

Cuenta con alrededor de 250.000 imágenes de distinta calidad visual, superando notablemente a sus predecesoras.

Además, su etiquetado es mucho más complejo y completo que el de estas, con características semánticas particulares de cada imagen acompañando al vector etiqueta con 210 votos de media.

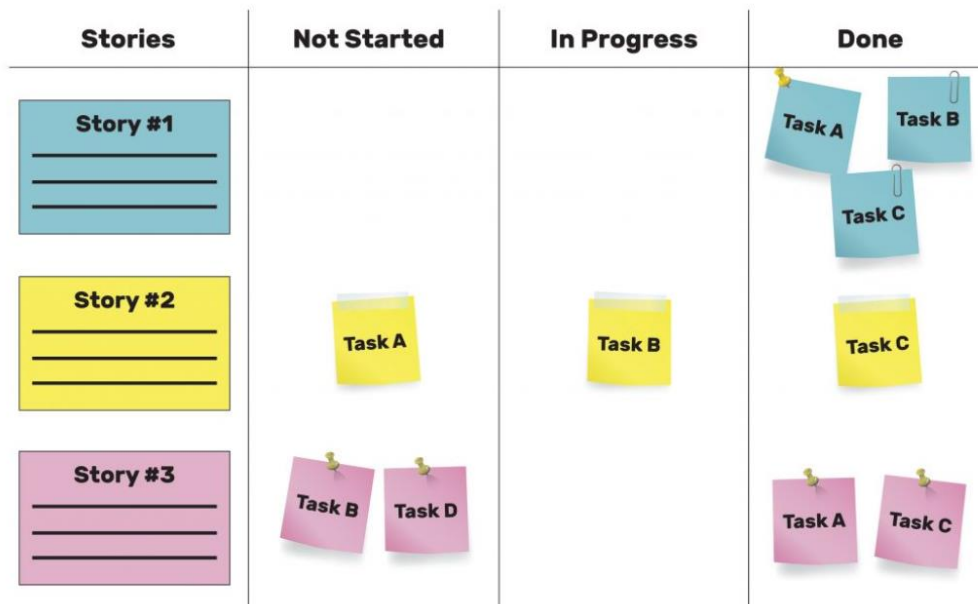
Toda la información que la compone fue recopilada de [www.dpchallenge.com](http://www.dpchallenge.com), un foro de retos fotográficos de distintos tipos y ámbitos. Cada fotografía postada en un reto es calificada por el resto de usuarios con una nota del 1 al 10. Así, AVA genera su vector etiqueta almacenando en posiciones de 0 a 9 el número total de votos recibidos de cada nota.

### 4.3 Metodología Agile SCRUM

Para el desarrollo de este proyecto decidí hacer uso de la metodología SCRUM que aprendimos en el paso por el grado.

Habitualmente, esta metodología ayuda al cliente a definir varias funcionalidades o aspectos deseados en su sistema de cara al SCRUM master o jefe del proyecto. Este se encargará de

dividir y organizar las tareas cronológicamente para poder alcanzar los subobjetivos fijados que logran los deseos del cliente.



**Figura 23:** SCRUM board

En mi caso, debo asumir los roles de todas las partes de esta metodología. Es decir, del cliente: por ser la parte interesada de que este proyecto triunfe; del SCRUM master: porque yo mismo me debo encargar de la gestión de mi proyecto; y del equipo de desarrollo: porque todo el desarrollo recae sobre mí mismo, también.

Por lo tanto y con el punto de mira puesto sobre el objetivo final del proyecto, clasificar estéticamente una imagen, decidí definir como primer subobjetivo o hito: construir modelos básicos de ambos tipos y probar su eficacia sobre CIFAR-10 ([Coronado, 2022](#)).

#### 4.3.1 1<sup>er</sup> Hito: Experimentos previos sobre CIFAR-10

### Basic Model Buildings

Closed 9 days ago 100% complete

This milestone is meant to set the deadline for us to acquire all the knowledge related with ViT and ConvMixers models and their structure.

Also we want to build, using this knowledge, a pair of basic models that will set the bases of our project.

Referenced papers:

- [Image classification with Vision Transformer](#)
- [Image classification with ConvMixer](#)

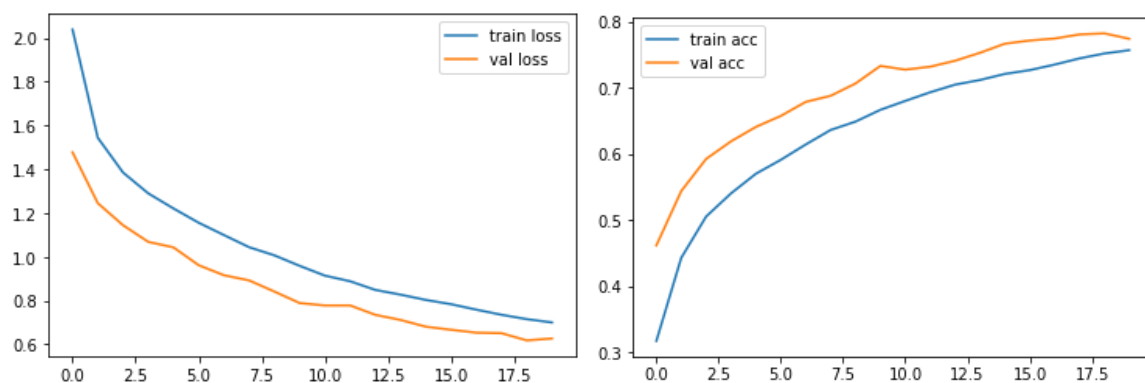
**Figura 24:** Hito 1 - gestor de proyectos GitHub

A partir de las estructuras teóricas que comentábamos en los apartados 3.2.4 y 3.3 construimos sobre un par de libretas python dos modelos que lanzaremos sobre el entorno de ejecución GPU Kaggle.

En el caso del ViT, fijamos los hiper-parámetros de entreno de la siguiente forma:

- learning\_rate: 0.001
- batch\_size: 256
- num\_epochs: 20

Obteniendo una precisión frente al conjunto de test del 77,4%. Y con la evolución en las métricas de pérdida y precisión según mostramos en las siguientes gráficas:



**Figura 25:** Evolución del modelo ViT sobre CIFAR-10

Por la forma que dibujan podemos deducir que el learning rate elegido ha sido bueno para el entrenamiento, siguiendo una evolución progresiva en la disminución del error y el aumento de la precisión.

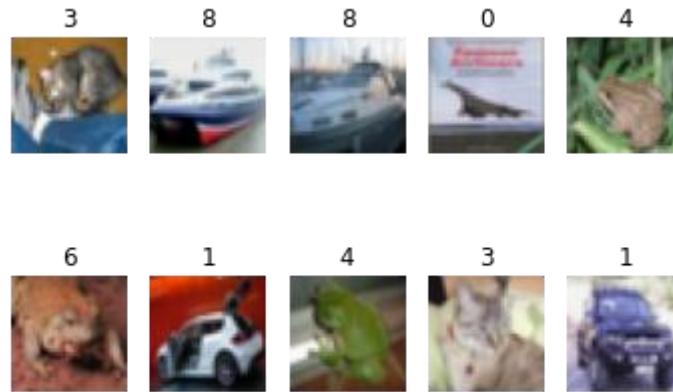
De forma adicional, quizá podríamos haber dejado a la red entrenar sobre unas pocas épocas más y conseguir resultados aún mejores.

Para el modelo ConvMixer fijamos los hiper-parámetros:

- learning\_rate: 0.001
- batch\_size: 128
- num\_epochs: 10

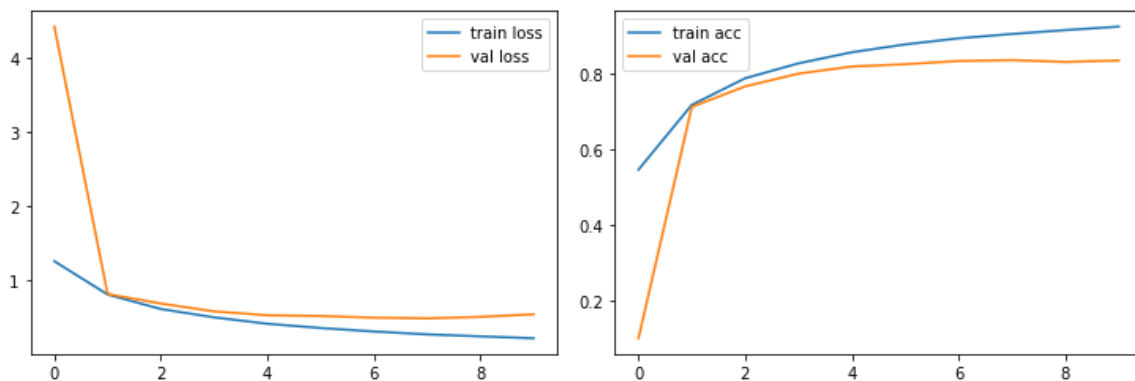
En este caso, y con un entrenamiento un poco más liviano, conseguimos resultados cercanos al 82% de precisión.





**Figura 26:** Demostración de la precisión del ConvMixer en CIFAR-10

Sin embargo, como vemos en la ilustración, probamos el modelo contra 10 imágenes aleatorias del conjunto de test y aun acertando correctamente en la mayoría de predicciones confunde dos ranas (etiqueta 6 del problema CIFAR-10) con ciervos (etiqueta 4).



**Figura 27:** Evolución del modelo ConvMixer sobre CIFAR-10

Visualizamos las gráficas de pérdida y precisión a lo largo del entrenamiento y en este caso podríamos decir que el learning rate es demasiado alto. Por tanto, para tantear su ajuste reduciríamos a  $10^{-4}$  o incluso  $10^{-5}$  y volveríamos a entrenar.

Sin embargo, ninguna de las dos mejoras las implementamos sobre las libretas ya que estos resultados son suficientes como indicativos de la potencia de ambas estructuras.

Ahora, el evolutivo del proyecto apuntará a pre-entrenar dichos modelos básicos sobre bases de datos de clasificación de imágenes aún más complejas, con la visión a futuro de utilizar los pesos internos de la red ya ajustados como punto de partida para el problema de AVA. A este proceso se le conoce como Transfer Learning. Y en nuestro caso, vamos a buscar modelos ViT y ConvMixer que hayan sido pre-entrenados durante un gran número de épocas contra ImageNet, de la que ya hablábamos en el apartado 3.1 ([Coronado, 2022](#)).

### 4.3.2 2º Hito: Experimentos previos de Transfer Learning

## Pre-trained transfer learning models

Closed 9 days ago 100% complete

This milestone is meant to set the deadline for us to create a competitive model for both the ViT and ConvMixer structures, as to solve accurately the classification problem CIFAR-100 defines.

**Figura 28:** Hito 2 - gestor de proyectos GitHub

Por las propiedades particulares de la base de datos objetivo, su tamaño y su carácter subjetivo, necesitamos un modelo ViT y ConvMixer competente para la Clasificación de Imágenes. Encontramos el repositorio GitHub [tfimm](#) que da acceso a modelos pre-entrenados en ImageNet. De entre los cuales tenemos los ViT y los ConvMixer.

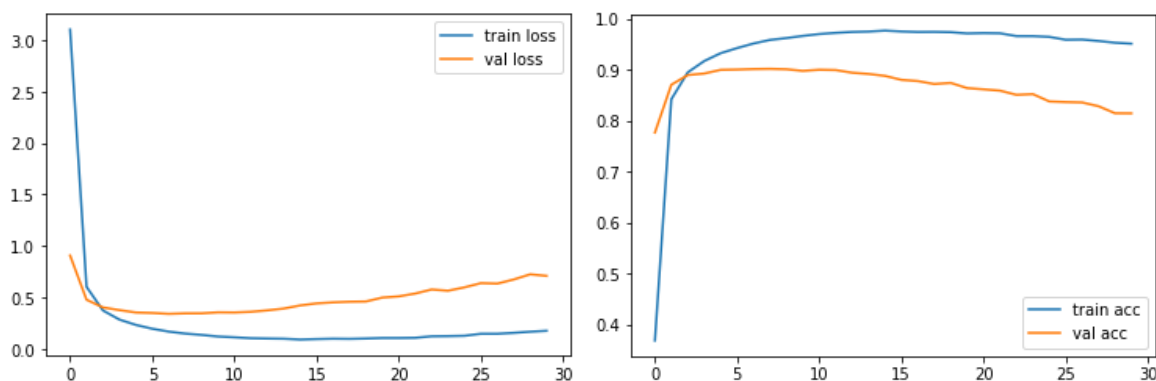
Por tanto y siguiendo las implementaciones del repositorio, generamos y probamos su eficacia contra una base de datos un poco más compleja, CIFAR-100.

Para el modelo ViT, cargamos el ViT-B/32 ([Dosovitskiy, 2020](#)) pre-entrenado sobre ImageNet que nos proporciona el framework tfimm y fijamos sus parámetros de entreno como sigue:

- learning\_rate: 0.00001
- batch\_size: 256
- num\_epochs: 30

El método de transfer learning que estamos siguiendo en estas libretas es el que conocemos como fine-tuning. Que consiste en descongelar por completo los pesos internos del modelo y re-entrenarlos con un learning rate muy bajo. Por ello, en este caso y casi obligatoriamente, debemos escoger un  $10^{-5}$  o incluso menor.

De cualquier forma, obtenemos una precisión del 90,57% bastante superior al 77,4% que veíamos en su versión básica.



**Figura 29:** Evolución del modelo ViT sobre CIFAR-100

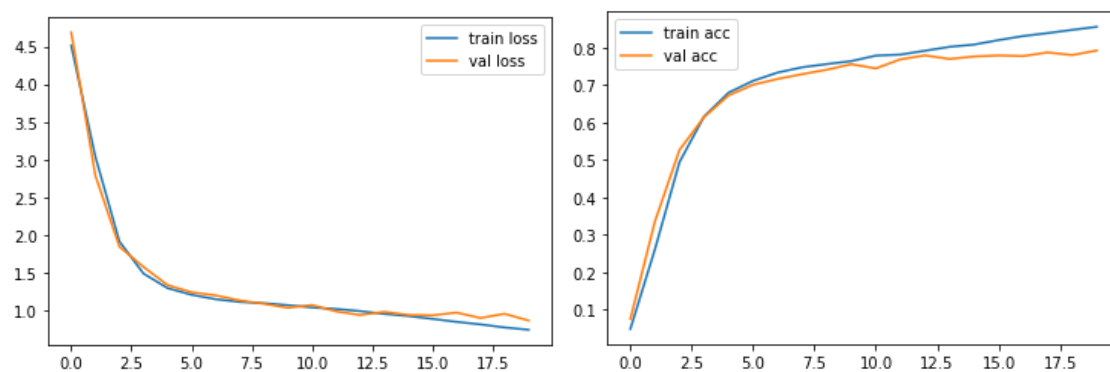
Las gráficas de precisión y pérdida nos muestran en este caso que con el paso de las épocas el modelo tiende a divergir, señal de que el learning rate sigue siendo demasiado alto. Como posible solución alternativa, podríamos valorar hacer uso de un early-stopper: hiper-parámetro que comentábamos en el apartado 2.2.2.1. y detiene la ejecución antes de que el modelo comience a divergir.

Igual que hacemos para el ViT, reducimos el learning rate de entrenamiento del ConvMixer quedando los hiper-parámetros ajustados de la siguiente forma:

- learning\_rate: 0.00001
- batch\_size: 32
- num\_epochs: 20

En este caso ya empiezan a notarse las limitaciones técnicas del entorno GPU Kaggle, con ejecuciones fallidas para tamaños de batch demasiado altos como 256 o 128. Tardando un total de 10 horas en terminar el entrenamiento que fijan los hiper-parámetros listados anteriormente.

Obtenemos una precisión en la clasificación de, recordemos, las 100 etiquetas de CIFAR-100 del 80,85% sobre el conjunto de test.



**Figura 30:** Evolución del modelo ConvMixer sobre CIFAR-100

Con una evolución de pérdida y precisión bastante buenas como indicativo de un ajuste correcto del learning rate. Pudiendo proponer, al igual que en el caso del ViT, únicamente como mejora el uso de un early-stopper que detuviera el entrenamiento en torno a la época 9.

### 4.3.3 3<sup>er</sup> Hito: Experimentos y resultados finales sobre AVA

Una vez vista la potencia del Transfer Learning a partir del uso de un modelo pre-entrenado en un problema parecido y de mayor escala que el objetivo, solo nos queda ponernos manos a la obra con el hito final. Tenemos un modelo pre-entrenado que teóricamente funciona bien en la clasificación de imágenes, solo nos queda entrenarlo para que lo haga estéticamente, también ([Coronado, 2022](#)).

## Aesthetic Visual Assesment Results

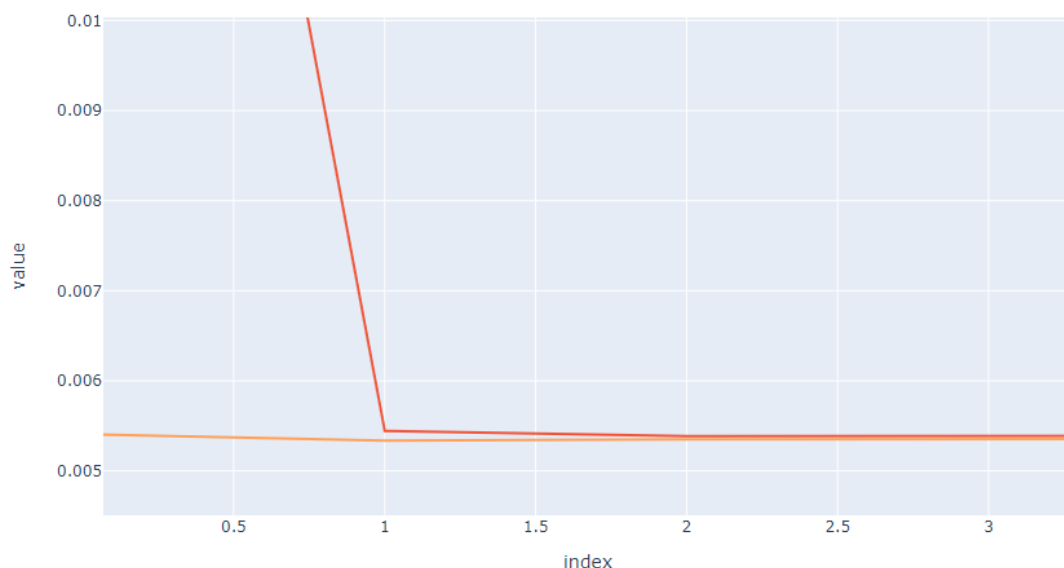
Closed 32 seconds ago 100% complete

This milestone is meant to set the final results of our project. We will need to test our pre-trained ([#2 Milestone](#)) models against the main goal dataset, AVA.

**Figura 31:** Hito 3 - gestor de proyectos GitHub

Hasta la fecha todas las ejecuciones las habíamos ido lanzando sobre el entorno online Kaggle. Ahora, por limitaciones técnicas del mismo, lo vamos a hacer en un entorno local Linux con una GPU NVIDIA-3080 instalada.

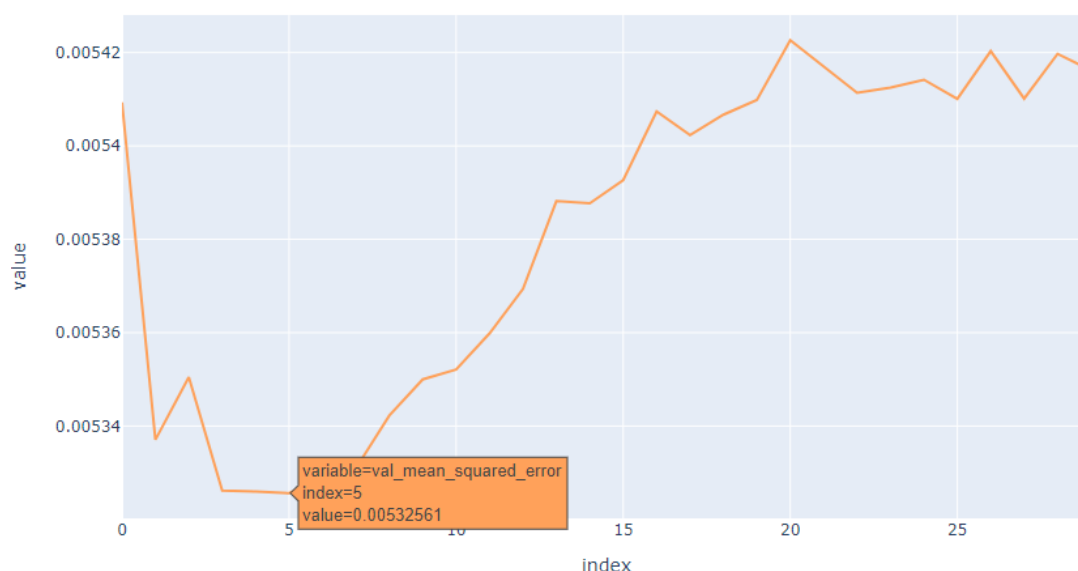
Para ello construimos la imagen Docker a partir de la [Kaggle v109](#) y un Dockerfile con los comandos `pip install` de `tfimm` y `timm` (repositorios con los modelos pre-entrenados). Y a través del repositorio *AQA-framework* del equipo de investigación del grupo SIMD, cargamos e indexamos sobre AVA.



**Figura 32:** Evolución del ECM en ViT sobre AVA

Obteniendo, sobre el ViT, un Error Cuadrático Medio de en torno al 0.0054, con una evolución en entrenamiento muy próxima a la final de validación. Y prácticamente convergiendo a partir de la tercera época.

Estos resultados tan positivos en épocas tan tempranas son claramente debidos a la potencia del pre-entrenamiento en ImageNet, que prepara al modelo para prácticamente cualquier problema de Visión Artificial.



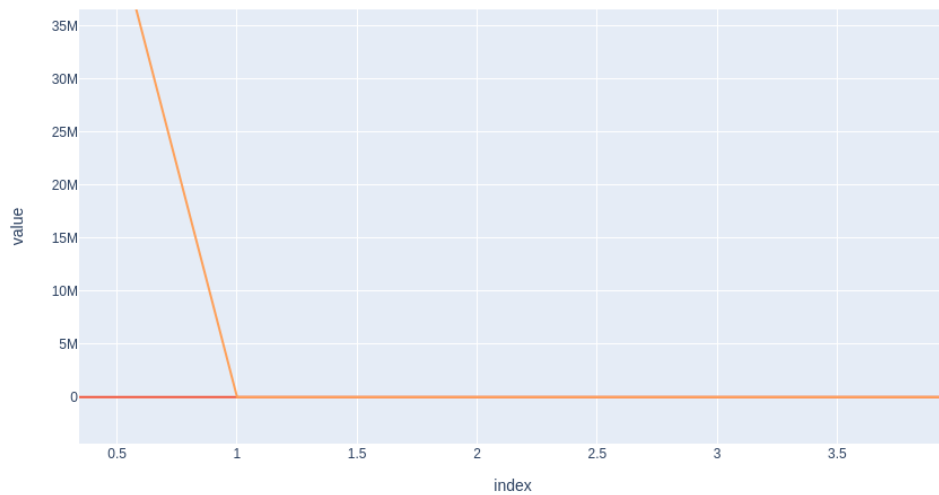
**Figura 33:** Evolución del ECM en ViT sobre AVA - Validación

Si analizamos la gráfica de validación por separado podemos ver como los resultados empiezan a empeorar a partir de la quinta época, donde alcanza su mínimo absoluto: 0.005325. Podríamos guardar, entonces, este modelo como el más competente de nuestro entreno y detener el resto de iteraciones ya que no nos van a aportar ninguna mejora.

Este experimento ha conllevado aproximadamente un total de 60 horas de ejecución, 2 horas por época. Reduciendo el tamaño de batch del anterior hito a 64 imágenes por conjunto y la tasa de aprendizaje a  $10^{-6}$ . Todo esto nos demuestra las grandes capacidades computacionales que requieren los entrenos y experimentos de Deep Learning.

Para el ConvMixer reducimos el tamaño de batch de 64 a 16 imágenes ya que el modelo realiza procesos computacionalmente más pesados que el Transformer y ocupa el total de la memoria GPU de nuestro equipo en la primera época. Además, por limitaciones en el tiempo real, debemos reducir el número de épocas de 30 a 20. Limitando así el total de nuestro experimento a 18 horas frente las 27 que suponen iterar 30.

Estas reducciones afectan a la eficiencia del entreno de nuestro modelo, pero no a la eficacia del modelo final.



**Figura 34:** Evolución del ECM en ConvMixer sobre AVA

Obteniendo resultados de en torno al 0.005 de Error Cuadrático Medio a partir de la segunda iteración. Y con unos resultados finales de 0.004914 y 0.055064 para el Error Cuadrático Medio y el Error Medio Absoluto, respectivamente.

En este caso, y al igual que en el ViT, los resultados nos muestran la gran potencia de los modelos pre-entrenados, alcanzando resultados competentes en épocas muy tempranas.

Todos los resultados e implementaciones de código de este e hitos anteriores se encuentran almacenados en el [Coronado, 2022](#).

# Capítulo 5

## Conclusiones y Trabajo Futuro

---

A lo largo del desarrollo del proyecto probamos estructuras profundas de relevancia contemporánea como los ViT y ConvMixer en bases de datos benchmark de su potencia como CIFAR-10 y 100. Obtuvimos resultados prometedores en ambos casos. Tanto para los modelos básicos, que conseguían precisiones de en torno al 80%, como para aquellos que pre-entrenábamos sobre ImageNet, con un 81% para el ConvMixer y cerca de un 91% para el ViT. Este segundo benchmark suponía limitaciones o hándicaps por su pequeño número de registros por etiqueta y, aun así, los modelos conseguían comportarse de forma certera en su clasificación.

Por último, probamos esa misma potencia sobre el problema objetivo del proyecto: la Evaluación Estética de Imágenes y comprobamos que de igual forma que ocurría en CIFAR-100 los modelos se comportan adecuadamente y nos reportan un error en sus predicciones cercano al 0.005, dando por alcanzado el objetivo de este trabajo.

Como ya hemos visto a lo largo del desarrollo de este proyecto, las principales limitaciones que encontramos en los problemas de Aprendizaje Profundo se deben a los requisitos computacionales que estos suponen.

De hecho, si analizamos las fuentes de los modelos pre-entrenados que usamos en los apartados 4.3.2 y 4.3.3, damos con Google. Que pre-entrena, por ejemplo, el [vit\\_base\\_patch32\\_224](#) con un tamaño de batch de 4096 imágenes, además de hacer uso de la tecnología TPuv3 de 8 núcleos. Por ello, no podemos comparar nuestros resultados con los que obtendríamos en un entorno tan preparado como el que ellos tienen. Con disponibilidad total en sus nodos y tiempo suficiente para poder entrenar modelos que lideran el Estado del Arte hoy en día.

Y es que no solo son importantes las especificaciones técnicas con las que cuentes como investigador y científico de Deep Learning. Una de las principales variables determinantes de que tu modelo termine siendo competente es el tiempo. Con un buen ajuste de hiper-parámetros puedes tener la confianza de dejar tu modelo entrenando días e incluso semanas, que acabará obteniendo resultados verdaderamente positivos.

Por ello, una de las posibles mejoras a futuro que se podrían hacer sobre esta línea de investigación es el ajuste de hiper-parámetros. Esta técnica está implementada sobre la librería Python *keras-tunner* compatible con *tensorflow*, proporcionándonos métodos basados en la Búsqueda Voraz de la mejor combinación de atributos.

Como ya vimos en las asignaturas de Diseño de Algoritmos y Metodología de Computadores, los algoritmos basados en la Búsqueda Voraz son muy costosos computacionalmente hablando. Si a esto le sumamos los tiempos de ejecución tan altos que suponen los entrenos individuales de modelos de Aprendizaje Profundo, se nos hace una práctica inasequible.

Por tanto y una vez más contando con los entornos de experimentación idílicos con los que cuenta Google, por ejemplo; podríamos hacer ajuste de hiper-parámetros y conseguir: más eficacia sobre el modelo final y más eficiencia sobre los entrenamientos del mismo. De hecho,



en el caso del [vit\\_base\\_patch32\\_224](#) de Google, pre-ajustan el hiper-parámetro más importante, el learning rate, con 10.000 iteraciones de Algoritmo Voraz.

En nuestro caso, sin hypertuning y tanteando el learning rate según vamos visualizando y analizando los resultados, obtenemos resultados sorprendentemente válidos.

De cualquier forma, poder probar la potencia de modelos que suponen el Estado del Arte actualmente en clasificación de imágenes contra problemas de carácter tan particular como la Evaluación Estética de Fotografías me parece una de las oportunidades más apasionantes que jamás me podría haber brindado el Departamento de Sistemas Inteligentes y Minería de Datos de la UCLM.

# Bibliografía

---

- Cavaioni, M. (2018, 25 junio). DeepLearning series: Convolutional Neural Networks - Machine Learning bites. *Medium*. <https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524> [9]
- Chamón, J. (2014, 2 diciembre). *Aplicación móvil para valoración automática de la calidad estética de fotografías*. RUIdeRA - UCLM. <https://ruidera.uclm.es/xmlui/handle/10578/4235> [7]
- Cooper, G. (2019, 3 octubre). New Vision Technologies For Real-World Applications. *Semiconductor Engineering*. <https://semiengineering.com/new-vision-technologies-for-real-world-applications/> [8]
- Coronado, D. (2022, 24 febrero). *Basic Model Buildings Milestone* · DaniCoronado/TFG. GitHub. <https://github.com/DaniCoronado/TFG> [17] [18] [20] [21]
- Dai, Z. (2021, 9 junio). *CoAtNet: Marrying Convolution and Attention for All Data Sizes*. arXiv.Org. <https://arxiv.org/abs/2106.04803v2> [12]
- Dosovitskiy, A. (2020, 22 octubre). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv.Org. <https://arxiv.org/abs/2010.11929v2> [15] [19]

- Gokila Brindha, P., Kavinraj, M., Manivasakam, P., & Prasanth, P. (2021, febrero). Brain tumor detection from MRI images using deep learning techniques. *IOP Conference Series: Materials Science and Engineering*, 012115. <https://doi.org/10.1088/1757-899x/1055/1/012115> [1]
- Graetz, F. M. (2021, 7 diciembre). How to visualize convolutional features in 40 lines of code. *Medium*. <https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code-70b7d87b0030> [10]
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), 2554–2558. <https://doi.org/10.1073/pnas.79.8.2554> [5]
- Iqbal, T., & Qureshi, S. (2020). The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2020.04.001> [2]
- Park, S. (2022, 4 enero). ConvMixer: Patches Are All You Need? Overview and thoughts [V]. *Medium*. <https://medium.com/codex/an-overview-on-convmixer-patches-are-all-you-need-8502a8d87011> [16]
- Quinlan, R. J. (1992). *C4.5: Programs for Machine Learning* (1. ed.). Morgan Kaufmann Publishers. [4]
- Ramesh, A. (2022, 13 abril). *Hierarchical Text-Conditional Image Generation with CLIP Latents*. arXiv.Org. <https://arxiv.org/abs/2204.06125> [3]
- Santana, C. [Dot CSV]. (2021, 27 septiembre). *Las REDES NEURONALES ahora prestan ATENCIÓN! - TRANSFORMERS ¿Cómo funcionan?* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=aL-EmKuB078&feature=youtu.be> [13] [14]

Yang, L., & Shami, A. (2020, noviembre). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 295–316.  
<https://doi.org/10.1016/j.neucom.2020.07.061> [6]

Zhai, X. (2021, 8 junio). *Scaling Vision Transformers*. arXiv.Org.  
<https://arxiv.org/abs/2106.04560> [11]