Universidad Zaragoza
1542

Trabajo Fin de Grado

Deep Learning en el arte y la ilustración
Deep Learning in art and illustration

Autor/es

Manuel Lagunas Arto

Director/es

Elena Garcés García

Ponente

Diego Gutiérrez Pérez

Escuela de Ingeniería y Arquitectura de Zaragoza

Noviembre de 2016

Graphics and Imaging Lab

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Manuel Lagunas Arto,

con nº de DNI 73025057B en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Deep Learning en el arte y la ilustración,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 29 de Agosto de 2016

Fdo: Manuel Lagunas Arto

Thanks Yana, thanks for being the partner of this trip called life, let's make it never end. My fiancée, thank you.

Thanks to my family, for all their endless support. Specially for you, dad, wherever you are, this is for you as well.

Thanks also to Elena and Diego for their help and guidance during the project and for their valuable feedback.

Debido a que este proyecto es de carácter exploratorio y de investigación, la mayor parte de la terminología relacionada procede del inglés. Por esta razón, muchos de los términos y expresiones técnicas más comunes carecen de traducción alguna al castellano o su traducción no es habitual en este contexto. Durante el desarrollo de este documento se ha tratado, en la medida de lo posible, de traducir la mayor parte de estos términos y expresiones, exceptuando casos en los que no existe traducción o su inclusión repetida en el texto afectaba a la fluidez de la lectura.

Deep Learning en el arte y la ilustración

El siguiente proyecto investiga las técnicas del aprendizaje profundo denominadas Deep Learning, que consisten en añadir un conjunto de capas no-lineales a las arquitecturas tradicionales de redes neuronales. Esta técnica consigue tener una mayor precisión y capacidad de abstracción para realizar tareas que, anteriormente, se asumían tan solo realizables con altos porcentajes de éxito por los humanos.

Más concretamente, en este proyecto se trata de comprender la forma de trabajo de las redes neuronales profundas, los algoritmos de optimización que emplean y su particular aplicación al problema de la clasificación de imágenes tanto naturales como artísticas. El problema de la clasificación de imágenes se trata de una tarea sencilla para el ser humano dada la experiencia adquirida con el paso de los años, pero realmente compleja de realizar por ordenadores, que deben traducir un conjunto de número (píxeles) en etiquetas con sentido semántico.

Además, se evaluarán las arquitecturas ya existentes en un nuevo dominio, las imágenes artísticas o ilustraciones, cuyas características a bajo y medio nivel difieren completamente de las imágenes naturales. Estas particularidades harán que su clasificación sea más compleja, sesgando las capacidades de abstracción de las redes pre entrenadas y, por tanto, obteniendo poco éxito con su uso.

Para obtener altos porcentajes de precisión se hará uso de la red como generador de descriptores de la clase, que van a ser clasificados usando una máquina de soporte de vectores. Además, con el fin de mejorar dichos resultados, se optimizarán los parámetros de la red de manera que los descriptores generados sean más precisos. Mejorando la precisión en hasta un 70 %.

Índice general

Índice de figuras

Página

Índice de tablas

Página

1 INTRODUCCIÓN

Con el paso del tiempo el ser humano ha adquirido la capacidad de poder identificar objetos, texturas y formas en las imágenes naturales sin requerir de un esfuerzo desmesurado. Dicha tarea, es mucho más complicada para los ordenadores que, para poder clasificar imágenes deben ser capaces de comprender su representación, que no deja de ser un conjunto de píxeles, de la cual deben obtener toda la información posible. Recientemente, estas tareas de clasificación de imágenes naturales han alcanzado niveles de acierto muy elevados gracias al uso del **aprendizaje profundo** (cuya traducción al inglés es Deep Learning), haciendo que su error sea menor de un 10 %. Antes de su aparición el estado del arte obtenía valores de alrededor de un 30 % utilizando algoritmos que consistían en un proceso de extracción características (SIFT y vectores de Fisher), con algún tipo de reducción de dimensionalidad como análisis principal de componentes (cuya traducción al inglés es PCA) [17] que, posteriormente, se clasificaban con máquinas de soporte de vectores (SVM) [14] [18].

El **aprendizaje profundo** hace referencia a un conjunto de técnicas de aprendizaje automático que emplean arquitecturas complejas con transformaciones **no lineales** para modelar abstracciones a alto nivel. Yann LeCun desarrolló una serie de arquitecturas utilizando redes neuronales con múltiples capas a finales del siglo XX [13] que dieron pie al deep learning, pero la baja potencia de las GPUs de entonces impedía el desarrollo de algoritmos eficientes y exitosos.

Actualmente, el aumento en la potencia de las tarjetas gráficas unido al aumento de la disponi-

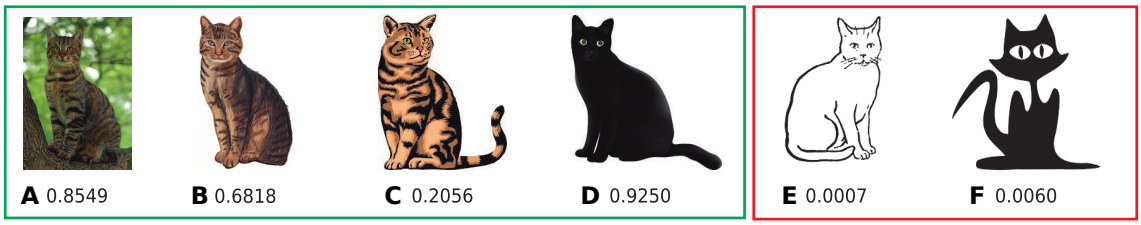


Figura 1.1: Comparativa de clasificación de imágenes naturales y cliparts con diferente grado de abstracción haciendo uso de la red VGG Net 19 mostrando la probabilidad entre 0 y 1 de que sea un gato. La imagen A se trata de una imagen natural, las imágenes de B a D corresponden a clip-arts con cierta similitud con una fotografía, mientras que los clip-arts E y F son completamente abstractos.

bilidad de datos etiquetados en la red con bases de datos como: ImageNet [5], CIFAR [10] o Pascal VOC [6] enfocadas a clasificación de imágenes; MNIST [12] enfocada al reconocimiento de dígitos manuscritos entre otras, han dado un gran empuje al uso de redes convolucionales profundas (las cuales obtienen los mejores porcentajes de error) y a la exploración de diferentes arquitecturas que estudian como la variación en **profundidad** [28] [23] y la **anchura** de las redes [27], o la **inicialización** de sus pesos [2] mejoran los porcentajes de precisión obtenidos.

Las redes neuronales convolucionales tienen un desempeño extraordinario clasificando imágenes naturales, además, se ha demostrado que pueden ser potentes descriptores de una clase [1] [16] si se extraen sus características a alto nivel y se entrena un clasificador con ellas [4]. Estas técnicas conocidas como **transferencias de conocimiento** son útiles para poder emplear la red con un nuevo tipo de dato similar al que se ha usado para su entrenamiento optimizando tan solo los parámetros de un número reducido de capas (las nuevas a añadir) obteniendo resultados notables. Sin embargo, si el dato a usar tiene unas características muy diferentes a los de entrenamiento de la red la transferencia de conocimiento puede fallar. Si se le une el hecho de usar tipos de imágenes ilustrativos o artísticos, con cierta componente de aleatoriedad donde las formas, colores o texturas de los objetos puedan ser completamente opuestas complica, aún más, la tarea de clasificación obteniendo resultados con altos porcentajes de error.

La Figura 1.1 contiene un conjunto de imágenes naturales y clip-arts clasificados haciendo uso de la red neuronal VGG Net 19, la cual obtiene resultados en el estado del arte en clasificación de imágenes naturales, pero, si el tipo de dato usado es de tipo artístico su precisión disminuye prácticamente,

como se ilustra. La figura muestra las probabilidades entre 0 y 1 de que cada imagen sea un gato. La imagen A se trata de una imagen natural que la red es capaz de clasificar sin problema alguno; las imágenes B, C y D son clip-arts que tratan de simular la realidad, a pesar de ello, la figura B no tiene una probabilidad de ser un gato que parezca determinante para la red y la figura C ha sido predicha con 0.39 como un tigre en vez de un gato, respecto a la imagen D, la red ha sido capaz de clasificarla sin problema alguno; ahora, si analizamos las imágenes con una mayor componente abstracta, se puede apreciar por la probabilidad de que sea un gato, que la red no es capaz de clasificarlo correctamente. Además, si obtenemos la mayor probabilidad devuelta se tiene que la figura E ha sido predicha como *sobre* y un 0.07 de probabilidad y la figura F ha sido clasificada como un *hacha* y probabilidad 0.08. Lo que indica que la red no solo no ha sabido clasificarlas correctamente, tampoco tiene seguridad de que la clase con mayor probabilidad devuelta sea la correcta.

Se ha visto cómo las redes neuronales profundas son capaces de clasificar imágenes naturales e imágenes artísticas realistas de manera precisa haciendo uso de técnicas de aprendizaje profundo, sin embargo, la clasificación de imágenes de tipo artístico con cierto grado de abstracción ha resultado ser insatisfactoria; por ello, nos preguntamos si: **es posible que una red neuronal aprenda el concepto de clase a alto nivel para, así, generalizar y reconocer objetos en cualquier tipo de representación pictórica.**

1.1. Objetivos

Este proyecto de fin de grado es de carácter exploratorio y de investigación, y tiene como objetivos: entender cómo funcionan las diferentes arquitecturas de red propuestas para esta tarea y poder diseñar nuevas arquitecturas que mejoren el funcionamiento de las anteriores. Concretamente, los objetivos finales son:

- Aplicar las arquitecturas conocidas para clasificar imágenes naturales o fotografías en un contexto completamente diferente como es el arte o la ilustración.
- Ser capaces de medir empíricamente el error y precisión de estas arquitecturas para tomar decisiones de diseño.
- Desarrollar nuevos modelos que consigan mejores resultados que los ya existentes para clasificar

imágenes artísticas o ilustraciones.

1.1.1. Tecnologías empleadas

Además de objetivos teóricos, la realización del proyecto también contiene objetivos prácticos como son la completa comprensión del framework a utilizar, en este caso, **Torch**[1] [3], desarrollado por el grupo de investigación en Inteligencia Artificial de Facebook, investigadores de Google DeepMind y Twitter. Se trata de un framework fácil de usar, comprensible y eficiente, desarrollado sobre LuaJIT y una sub-implementación en C/CUDA que permite hacer uso de la potencia que nos brindan las GPU. LuaJIT es un compilador *Just-In-Time* (JIT) para Lua, un lenguaje de programación interpretado con sintaxis simple y con base en C y Perl. El framework Torch fue usado hasta hace unos meses por Google DeepMind, en su famoso programa AlphaGo [22], que fue capaz de derrotar al campeón mundial de Go por primera vez en la historia.

Conjuntamente a Torch se emplea **Python** en su versión 2.7 con el paquete *sklearn*, que será útil para el entrenamiento y evaluación de las máquinas de soportes de vectores (SVM). La comunicación entre Torch y Python se establece mediante ficheros con formato *HDF5* que permiten un fácil almacenamiento y una sencilla manipulación de grandes matrices. Conjuntamente a las tecnologías mencionadas anteriormente, se hace uso de **GitHub** una plataforma online que permite llevar un control de versiones, donde se encuentra alojado un repositorio con el proyecto[2] y otro repositorio con la memoria[3]. Se puede encontrar una explicación más detallada de las tecnologías en el Anexo B

1.2. Entorno del proyecto

El proyecto ha sido desarrollado en el grupo de investigación Graphics and Imaging Lab (**GILab**), perteneciente al Departamento de Informática e Ingeniería de Sistemas (DIIS) en la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza (EINA). Ha sido dirigido por Elena Garcés García y supervisado por el Dr. Diego Gutiérrez Pérez.

[1] Proyecto Torch: http://torch.ch/

[2] Página del proyecto: https://github.com/mlagunas/DLart

[3] Página de la memoria https://github.com/mlagunas/DLart_projectReport

1.3. Estructura de la memoria

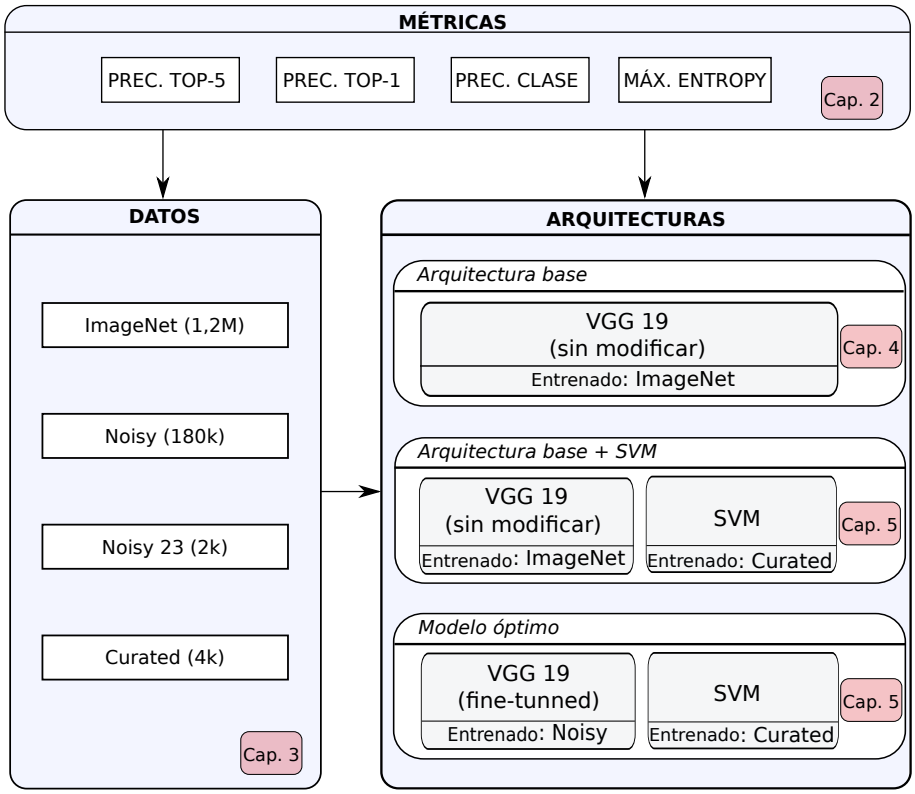


Figura 1.2: Vista general de las componentes del proyecto realizado con sus capítulos asociados.

En la Figura 1.2 se puede observar un esquema del proyecto realizado. De manera transversal se pueden ver las *Métricas* que, aplicadas a los datos, se utilizaron para obtener las mejores clases y un conjunto válido más exhaustivo. Además, las métricas se usaron en cada arquitectura propuesta para evaluar su desempeño y poder realizar comparaciones objetivas entre ellas. Los *Datos* contienen los conjuntos de imágenes, tanto naturales como ilustraciones, empleados en el trabajo. Por último, el grupo de las *Arquitecturas* comenta la red neuronal usada, así como todos los cambios realizados en esta para conseguir mejores porcentajes de precisión en la clasificación de imágenes.

Capítulo 2: Explica el conjunto de métricas empleadas: precisión top-1 y top-5 y la función de máxima entropía, que será también la función de error para el entrenamiento de la red neuronal profunda. Además, se explora el estado del arte actual en materias como: la clasificación de imágenes y la transferencia de conocimiento en redes neuronales.

Capítulo 3: Contiene la descripción de todos los conjuntos de datos empleados en este trabajo. Se comienza comentando ImageNet (I), el conjunto de imágenes naturales. Posteriormente, se detalla el proceso de obtención de los tres nuevos conjuntos de imágenes de clip-art: Noisy (N), Noisy 23 (N23) y Curated (C), su número de clases y entradas, sus problemas e inconvenientes, y las soluciones propuestas.

Capítulo 4: Comenta la arquitectura base empleada, que consiste en la red neuronal VGG Net 19, pre-entrenada con el conjunto de imágenes ImageNet. Se verán además los resultados obtenidos al tratar de emplear dicha arquitectura en un contexto diferente al que se entrenó, como son los clip-arts, y las conclusiones que se han obtenido.

Capítulo 5: Explica los dos nuevos modelos propuestos basados en la arquitectura base cuyo fin es mejorar la precisión final obtenida en la clasificación de los nuevos conjuntos de imágenes. Introduce una primera mejora del modelo inicial donde se hace uso de una máquina de soporte de vectores (SVM). Posteriormente, se describen todos pasos realizados hasta obtener la segunda arquitectura propuesta, denominada el modelo óptimo.

Capítulo 6: Valoración y comentarios acerca de la investigación realizada, conclusiones, ideas de aplicación de la tecnología desarrollada y posibles futuras líneas de investigación.

2 CONTEXTO TECNOLÓGICO

El siguiente capítulo contiene dos secciones de carácter descriptivo con información necesaria para la posterior comprensión del documento. Más concretamente, la Sección 2.1 explica las métricas usadas para comparar los algoritmos propuestos usando técnicas de Deep Learning, posteriormente, la Sección 2.2 es una exposición de las investigaciones que dieron pie al aprendizaje profundo, la evolución en las arquitecturas usadas para la clasificación de imágenes y los métodos empleados para obtener descriptores de los objetos a partir dichas arquitecturas con el fin de realizar transferencias de conocimiento en una misma red neuronal. Además, en el Anexo A se encuentra una detallada introducción al Deep Learning desde los conceptos más básicos.

2.1. Métricas empleadas

La tarea de clasificación de imágenes consiste en dar una etiqueta o clase a un dato de entrada. El hecho de dar a cada imagen una sola clase podría generar problemas de ambigüedad si en dicha entrada se pudiera encontrar más de un objeto. Por norma general, las métricas usadas para evaluar estas tareas proponen considerar las cinco primeras clases devueltas con mayor probabilidad por la red neuronal sin penalización siempre que en ellas se encuentre la clase real de la imagen. Para evaluar cómo actúan los distintos modelos se hace uso de una métrica que será la **precisión top-5** donde

cada imagen i tiene una sola clase C_i, la arquitectura propuesta podrá devolver hasta cinco clases $c_{i1}, \ldots, c_{i5}$ que serán correctas si $c_{ij} = C_i$. La precisión se calculará $d_{ij} = d(c_{ij}, C_i) = 1$ si $c_{ij} = C_i$, y 0 en cualquier otro caso. La precisión global será la suma del mayor número posible de aciertos ($d_{ij} = 1$) devueltos para el conjunto de imágenes, considerando n el número total de imágenes:

$$prec_5 = \frac{1}{n} \sum_{i}^{n} \max_j(d_{ij}) \tag{2.1}$$

Dado que el conjunto de imágenes también ha sido curado y muchas de sus entradas carecen de ningún tipo de ambigüedad posible, es necesario poder evaluar cómo actúan considerando tan solo la primera predicción devuelta (la de mayor probabilidad), para ello se modifica la **precisión top-5** para considerar tan solo el primer resultado obtenido, llamándolo **precisión top-1**. La precisión será ahora $d_{i1} = d(c_{i1}, C_i) = 1$ si $c_{i1} = C_i$, y 0 en cualquier otro caso, obteniendo un resultado global donde no es necesario obtener el máximo j, pues solo hay una clase posible, la primera.

$$prec_1 = \frac{1}{n} \sum_{i}^{n} (d_{i1}) \tag{2.2}$$

Además de globalmente, para poder evaluar de manera precisa una red, será necesario ver su desempeño por clase, lo que permite diagnosticar posibles problemas durante el entrenamiento. Para ello, se modifica el valor n de la fórmula de la precisión global, para que el conjunto evaluado sea solo el de las imágenes pertenecientes a la clase l, siendo el número de imágenes de este conjunto n_l:

$$prec_{1l} = \frac{1}{n_l} \sum_{i}^{n_l} (d_{i1}) \tag{2.3}$$

$$prec_{5l} = \frac{1}{n_l} \sum_{i}^{n_l} \max_j(d_{ij}) \tag{2.4}$$

Estas medidas serán útiles para comparar modelos, y su desempeño global y específico. Como medida adicional se hace uso del **error de máxima entropía**, que evalúa la seguridad de la arquitectura midiendo la probabilidad de la clase real de la imagen, es decir, relacionando las probabilidades (S) con las etiquetas reales (L) y obteniendo su valor medio para todas las imágenes del conjunto n. De nuevo, podrá usarse de manera individual para cada clase l si solo se consideran sus etiquetas e imágenes, teniendo ahora el conjunto de imágenes n_l. Explicada en detalle en el Anexo A.1.3.

$$D(S, L) = -\sum_{i}^{n} \log(S_i) * L_i \tag{2.5}$$

$$D(S, L)_l = -\sum_{i}^{n_l} \log(S_i) * L_i \tag{2.6}$$

2.2. Estado del arte

Hasta finales del siglo XX, la mayoría de técnicas que hacían uso de aprendizaje automático usaban arquitecturas *shallow-structured*, como pueden ser las máquinas de soporte de vectores (cuya traducción es *Support Vector Machines*,SVMs) o los perceptrones multicapa, (MLPs) entre otros. Estas arquitecturas demostraron ser capaces de resolver un gran conjunto de problemas de manera eficiente pero sus restricciones a la hora de modelarlos y su bajo poder de representación hacen que sean débiles a la hora de lidiar con problemas cotidianos más complejos.

La aparición del Deep Learning, como una variación de los MLPs con un gran número de capas ocultas, donde existen capas no lineales, permitió que muchos problemas hasta entonces impensables de resolver por máquinas tuvieran soluciones con grandes tasas de acierto. LeCun en 2001 presentó un trabajo de investigación donde el Deep Learning comenzaba a resaltar en la tarea del reconocimiento de documentos con la arquitectura *LeNet* [13] una red que hace uso de 7 capas con pesos. En dicho trabajo muestra los primeros pasos para confiar más en arquitecturas de aprendizaje automático que en heurísticas creadas ad-hoc para cada tipo de problema.

Posteriormente, dada la mayor cantidad de datos disponibles y el mayor número de imágenes etiquetadas comenzaron a desarrollarse nuevas arquitecturas más profundas, con mayor precisión y capacidad de representación. Comenzaron a aparecer competiciones como **ILSVRC** (ImageNet Large Scale Visual Recognition Challenge) [20] [5] con más de 1 millón de imágenes etiquetadas para poder entrenar arquitecturas de red profundas que las clasifiquen en 1000 clases diferentes. En 2012, **Alex Krizhevsky** [11] propuso una arquitectura prometedora mejorando los resultados anteriores en esta competición, donde además, desarrolló una nueva implementación de convolución 2D mucho más eficiente que hacía posible el entrenamiento con gran cantidad de datos. Dicha arquitectura se trata de una red convolucional profunda con 5 capas de convolución con un tamaño de ventana de 11 × 11 y 3 capas fully-connected que formarán el clasificador lineal. La red hace uso de ReLus [15] y de dropout [24] para evitar el overfitting dada su profundidad.

Durante los años siguientes los ganadores han sido arquitecturas basadas en la propuesta por

Krizhevsky, en el año 2013 Zeiler y Fergus [28] proponen una técnica de **visualización de activaciones** en las capas intermedias haciendo uso de redes desconvolucionales o *deconvolutional neural networks* pudiendo observar que la red en sus capas bajas era capaz de reconocer características de la imagen a bajo nivel, mientras que a alto nivel es capaz de capturar objetos completos.

El Visual Geometry Group de Oxford presentó al ILSVRC de 2014 una arquitectura llamada VGG Net [23]. En ella exploran cómo afecta la profundidad de la red al resultado obtenido, obteniendo mejores porcentajes de error que en anteriores casos. El problema que podría acarrear sería los tiempos necesitados para llevar a cabo el entrenamiento. Por ello redujeron el tamaño de los filtros y la ventana a 3×3 píxeles con stride y padding de un pixel. Su mejor resultado es un error top-5 de $7,3\,\%$ en el conjunto test del ILSVRC.

Oquab et. al. [16] presentaban una solución al problema de **transferencia de representaciones**, donde, haciendo uso de una red neuronal convolucional profunda ya entrenada para el dataset ImageNet eran capaces de obtener porcentajes de error relativamente bajos reiniciando la última capa fully-connected del clasificador lineal y añadiendo dos nuevas que serán reentrenadas sobre el conjunto de imágenes del nuevo problema. Más cercano al trabajo propuesto, Crowley y Zisserman [4], del Visual Geometry Group de Oxford, en la misma línea de transferir el conocimiento para poder usado en otro tipo de problema reutilizando una red ya entrenada, proponen el uso de **máquinas de soporte de vectores** (SVM) para clasificar las activaciones obtenidas en las últimas capas de una red neuronal convolucional. Las SVMs serán entrenados haciendo uso del nuevo conjunto de imágenes, que en este caso es de cuadros. Babenko et. al. [1], demuestran en su trabajo que las capas superiores de las redes convolucionales profundas pueden funcionar bien como **descriptores del contenido visual** de la imagen (llamado *neural codes*). Haciendo uso de *Principal Component Analysis* (*PCA*) [17] sobre los neural codes obtenidos de una red entrenada con ImageNet en la última capa convolucional y la primera fully-connected son capaces de obtener prometedores resultados en la tarea de recuperación de imágenes para distintos conjuntos. Sharif et. al. [19] reforzaba la teoría de la transferencia de contenido gracias a SVMs para clasificar las últimas capas haciendo uso de la red *OverFeat* [21], una red convolucional profunda entrenada con ImageNet, con la que obtenía resultados similares al estado del arte en otros conjuntos diferentes al usado para entrenarla.

Dada la capacidad de las redes convolucionales profundas para sobrepasar problemas a la hora de

la transferencia de conocimiento [4], en el siguiente trabajo de investigación se hará uso de la red VGG Net con 19 capas con pesos. Además, se hará uso de las activaciones de las últimas capas de la red [1], más concretamente la segunda capa fully-connected, para que, posteriormente sean clasificadas por una SVM, tal y como sugieren el trabajo de Crowley y Zisserman [4], y Sharif et. al. [19]. Los primeros hacen uso de un conjunto de cuadros con complejidades relativas al estilo o el tiempo en el que fueron pintados y los segundos usan distintos conjuntos de imágenes naturales. Por el contrario, nuestro modelo hace uso de imágenes de clip-art cuya complejidad reside en las características de la imagen a bajo nivel: bordes, colores, etc. También se hará una optimización de la red, reentrenando de manera selectiva las capas que generan errores con las imágenes de entrada, de manera que las activaciones finales sean más restrictivas y precisas a la hora de actuar como descriptores de una clase que serán, posteriormente, clasificados con una SVM.

3 OBTENER LOS DATOS

La cantidad de información etiquetada en la red ha crecido de manera exponencial, haciendo fácil su obtención para el posterior entreno de las redes neuronales profundas. Dicho crecimiento no ha sido igual con todos los tipos de imagen focalizándose en las fotografías, haciendo que otros conjuntos de imágenes no existan o sean muy limitados como conjuntos de datos de cuadros.

Por tanto, en este proyecto se ha tenido que obtener un conjunto de imágenes válido requiriendo un tiempo de procesamiento, limpieza y curado de los datos antes de que estos puedan ser usados. Este capítulo comenta en la Sección 3.1 la conocida base de datos ImageNet, las imágenes que contiene, así como la competición que organiza cada año, posteriormente, en la Sección 3.2 se comenta el conjunto de imágenes de ilustración propio, como se ha obtenido y el procesamiento necesario para que fuera válido.

3.1. Base de datos ImageNet

ImageNet (I) [5][1] es un conjunto de datos con más de 15 millones de imágenes de alta resolución que pertenecen a alrededor de 22,000 categorías. Estas imágenes fueron recogidas de páginas web

[1] Web de ImageNet http://image-net.org/

como *flickr* y etiquetadas manualmente haciendo uso de la herramienta de Amazon: Mechanical Turk, una herramienta de *crowd-sourcing*. En el año 2010, comenzaron una competición anual llamada *ImageNet Large Scale Visual Recoginition Challenge* (**ILSVRC**) [20] que hace uso de un subconjunto de ImageNet con 1,2 millones de imágenes distribuidas en 1,000 clases. De ellas, alrededor de 1,000,000 son para entrenamiento, 50,000 para validación y 150,000 para test obtenidas independientemente.

Figura 3.1: Muestra de parte de las imágenes de la base de datos ImageNet.

El **ILSVRC** es una competición que ofrece un gran número de imágenes etiquetadas cuyo objetivo es poder evaluar el desarrollo de los algoritmos propuestos para generar un etiquetado automático, además de, ofrecer a la comunidad científica una base de imágenes amplia para poder probar sus modelos. La competición ofrece 3 conjuntos de imágenes para cada una de sus sub competiciones: entrenamiento y validación que disponen de las etiquetas y test, el cual solo contiene las imágenes y para el cual cada algoritmo propuesto debe ser el encargado de generar el etiquetado correspondiente. Las imágenes de test se obtienen de manera independiente a las que aparecen en ImageNet y por tanto no es posible conocer su etiqueta real. ILSVRC tiene 3 sub competiciones: clasificación de

imágenes, localización single-object, y detección de objetos. El interés de esta competición para el trabajo desarrollado reside en las clases que proponen para la tarea de clasificación de imágenes. A la hora de obtener nuevos conjuntos de datos válidos, se realizará un mapeo a las 1000 clases que ofrece dicha competición, de manera que una red neuronal pre-entrenada con los datos ofrecidos por el **ILSVRC** sea usable con los nuevos datos obtenidos.

3.2. Conjunto de imágenes de Clip-art

El trabajo de investigación desarrollado busca clasificar imágenes con el menor error posible, dichas imágenes deben ser ilustraciones o de tipo artístico. Para la obtención de un conjunto de imágenes válido se ha seguido el proceso representado en la Figura 3.2. Se ha hecho uso de una notación similar a $*1$ para poder tener correspondencia entre el texto y cada parte del proceso representado en el gráfico.

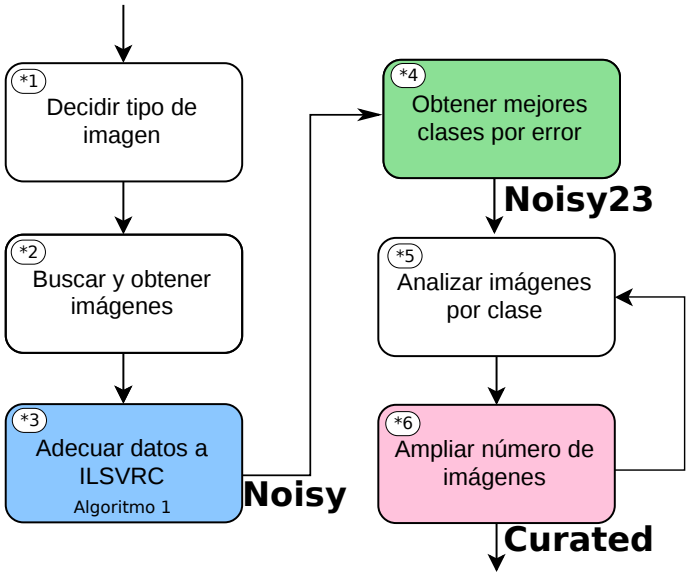


Figura 3.2: Pasos a realizar hasta obtener un conjunto de datos usable para una red neuronal profunda.

Para este proyecto, se decidió hacer uso de **clip-arts**[*1] dado su carácter artístico, su disponibilidad en la red y complejidad a bajo nivel que hacían que lograr el objetivo propuesto fuera un reto (ver Figura 3.3). Tras conocer el tipo de imagen a usar, el siguiente paso a realizar fue **obtener un número de imágenes que fuera considerable**[*2] de manera que la red neuronal a usar se pudiera optimizar para el nuevo tipo de dato. Para ello se hizo uso del conjunto de imágenes de clip-art usado en

la investigación de Garces et al. [7] que buscaba una métrica de similitud de estilos entre distintas imágenes de este tipo. El conjunto de datos contiene más de 200,000 entradas de tamaño medio 400 × 400 píxeles, etiquetadas por su significado semántico, no guardando correspondencia con las 1000 clases que da la competición ILSVRC.

Figura 3.3: Muestra de parte de las imágenes de clip-art obtenidas.

Dado que el etiquetado no corresponde con el del ILSVRC, no existe manera posible de poder evaluar cuantitativamente el desempeño de una red neuronal profunda sobre este conjunto de datos. Por ello es necesario realizar un pre proceso que descarte las imágenes que no tienen representación dentro de estas 1000 clases y empareje el resto con sus posibles etiquetas*[3]. El proceso comienza obteniendo todas las entradas del conjunto de datos y las 1000 clases de la competición, separando sus nombres por palabras y eliminando *stop-words*. Posteriormente se hace un recorrido de manera que si una palabra dentro del nombre de una imagen corresponde a una de las palabras de una clase esta se copia a la ruta *clase/imagen.png*, por ejemplo: la imagen *terrier-1.png* se copiará en la ruta *terrier/terrier-1.png*, por lo tanto, puede haber varias imágenes iguales en clases diferentes. Este proceso puede verse en detalle en el Algoritmo 1. Tras finalizar este proceso, el resultado es un nuevo conjunto de imágenes *clip-art* con más de 180,000 entradas distribuidas en un total de 826 clases. Este conjunto de datos recibe el nombre de **noisy**.

Algorithm 1 Adecuar datos a ILSVRC.

```
procedure Adecuar_conjunto(Vector imagenes, Vector clases)
    for cada clase c en clases do
        clase ← nombre de la clase
        vector_clase ← separar palabras y eliminar stopwords
        matriz_clases ← añadir vector_clase
    for cada imagen i en imagenes do
        nombre ← nombre de la imagen
        vector_nombre ← separar palabras y eliminar stopwords
        for cada vector v_c en matriz_clases do
            for cada palabra p_c en v_c do
                for cada palabra p_i en vector_nombre do
                    if p_c = p_i then copiar imagen p_i en carpeta de clase p_c.
```

3.2.1. Curando y poblando el conjunto de datos

Dado que a la hora de crear el conjunto de datos noisy se emplea una restricción débil (cuyo objetivo es obtener el mayor número de imágenes el dataset) que puede generar problemas de ambigüedades en clases con nombres compuestos, por ejemplo una imagen de un perro, *dog.png* podría estar en la clase *hot-dog*, problemas de sinonimia donde la clase *crane* puede significar grúa y grulla haciendo que contenga elementos de ambos e incluso problemas en el número de ejemplos, existiendo clases con tan solo una imagen o clases con más de 1500 entradas.

Por ello, se hace uso de la red neuronal VGG Net 19 [23] sin ninguna modificación en su arquitectura y con los mismos pesos y bias que tiene tras su entrenamiento con ImageNet. La red sirve para hacer una clasificación del conjunto de imágenes de clip-art completo obteniendo la precisión top-1 y top-5, y el error de la función de máxima entropía. De acuerdo a estos resultados se seleccionan las **mejores clases**[*4] que no presenten además posibles problemas de ambigüedad ni de sinonimia (clases con nombres de una sola palabra) y que tienen un número de imágenes representativo, es decir, más de 25. Obteniendo un total de **23 clases finales**, las cuales están representadas en la Tabla 3.1.

Dichas clases, a pesar de ser las que mejores resultados tienen al ser clasificadas, siguen teniendo

Dataset	Ambulance	Banjo	Cassete	Desk	Envelope	Goblet
Noisy	35	26	62	400	43	27
Curated	144	122	134	458	132	135
Dataset	Hammer	Harp	Hourglass	Jellyfish	Mask	Mosque
Noisy	125	45	46	27	224	26
Curated	176	127	135	115	278	120
Dataset	Pelican	Printer	Shovel	Stove	Syringe	Teapot
Noisy	59	153	95	51	51	89
Curated	150	235	155	174	152	199
Dataset	Toaster	Trombone	Umbrella	Vase	Zebra	global
Noisy	70	32	203	157	31	2077
Curated	185	141	254	244	126	4091

Tabla 3.1: Comparativa de los resultados finales sobre el número de imágenes del conjunto de datos "curated" y el conjunto de datos "noisy" con las 23 clases usadas en ambos conjuntos.

los problemas que tenía el conjunto de datos noisy, por ello, habrá que realizar una **limpieza manual**[*5] eliminando imágenes que no tuvieran correspondencia con la clase en cuestión, obteniendo, finalmente, un conjunto de datos curado, pero sin ejemplos suficientes en algunas de las clases. Se hace uso de *web-scrapping* de *Google Images* [2] para obtener un mayor número de ejemplos por clase. Para ello se crea una consulta con el nombre de la clase y se establece el tipo de imagen para la búsqueda en clip-art. Dicha consulta construye urls que haciendo uso de Firefox descargan un número de entradas predeterminado, en este caso serán 200 nuevas imágenes por clase[*6]. Dichas imágenes al estar descargadas automáticamente no tienen por qué corresponder con la clase en cuestión, por tanto, posteriormente se curarán manualmente eliminando posibles datos erróneos que no correspondieran a su etiquetado. Como resultado final se obtiene un conjunto de imágenes de clip-art el cual recibe el nombre **curated** con 4091 imágenes distribuidas en 23 clases. Para que los futuros resultados a obtener tengan relación y sean coherentes con las 23 clases del conjunto curated, habrá que obtener

[2]Código obtenido de: https://github.com/shuvronewscred

por separado dichas clases del conjunto de imágenes noisy a la hora de obtener comparativas y estadísticas.

Tras finalizar el proceso de obtención y curado de los datos se obtienen 3 conjuntos de imágenes:

Imagenet (I) Conjunto de datos correspondiente a la competición ILSVRC. Contiene 1,2 millones de imágenes separadas en: 1.000.000 para entrenamiento, 50.000 para validación y 150.000 para test. En el correspondiente trabajo se hace uso de la red pre-entrenada con ese 1.000.000 de datos y se usa el sub-conjunto de validación para evaluar el desempeño de las arquitecturas desarrolladas.

Noisy (N) Contiene las imágenes directamente extraídas del conjunto obtenido por Garces et al. [7] tras realizar el mapeo de su etiquetado semántico a las clases del ILSVRC. Tiene problemas de ambigüedad en sus clases. Contiene 23 clases y más de 180,000 imágenes. Es útil para medir el desempeño global de la red.

Noisy 23 (N23) Este conjunto de imágenes corresponde con el conjunto Noisy pero ha sido reducido a las 23 clases del conjunto curated para poder realizar evaluaciones y comparaciones más precisas entre ambos grupos de imágenes. Contiene 2077 imágenes entre sus 23 clases.

Curated (C) Contiene las 23 clases del conjunto noisy con un mayor porcentaje de precisión top-1 y top-5 así como menor error de máxima entropía. Además, cada una de estas clases tiene un número mínimo de 25 imágenes. Cada clase ha sido curada manualmente eliminando posibles incoherencias entre la imagen y el etiquetado dado. Finalmente se han añadido nuevas entradas a cada clase haciendo web-scrapping de Google images teniendo 4091 imágenes totales.

4 REDES NEURONALES PROFUNDAS

Tras obtener un conjunto de datos válido, el siguiente objetivo es conseguir una red neuronal profunda previamente entrenada sobre imágenes naturales, para medir su desempeño con clip-arts, y comprobar si es capaz de abstraerse lo suficiente como para llegar a reconocer este tipo de dato nuevo para la red. Para ello, se hará uso de la red VGG Net 19 explicada en detalle en la Sección 4.1, capaz de obtener los resultados más avanzados en clasificación de imágenes. En la Sección 4.2 se describen las pruebas realizadas para comprobar la capacidad de abstracción de la red sobre los conjuntos de imágenes de clip-art creados haciendo uso de la precisión top-1 y top-5, además del error de máxima entropía.

4.1. La red VGG Net 19

Los buenos resultados conseguidos en la competición ILSVRC [20] haciendo uso de redes convolucionales [11] [28] y la potencia de las nuevas tarjetas gráficas de entonces permitieron a Simonyan y Zisserman [23] desarrollar un conjunto de **arquitecturas más profundas y precisas**, obteniendo unos de los porcentajes de error top-5 y top-1 más bajos en clasificación de imágenes y localización de objetos dentro de la competición.

La entrada de las redes convolucionales (tipo de red neuronal explicado en el Anexo A.3) pro-

puestas por Simonyan y Zisserman es una imagen de 224×224 de 3 canales con su valor RGB medio sustraído de cada pixel. La Figura 4.1 muestra la arquitectura de la red de 19 capas con pesos, Las convoluciones tienen un tamaño de filtro muy pequeño, 3×3, necesario para capturar la noción de arriba/abajo y derecha/izquierda con stride y padding de 1 pixel evitando perder dimensionalidad. La red contiene 5 capas de pooling con la función *max*, un tamaño de ventana de 2×2 y stride de 2 píxeles. Tras las capas de convolución y pooling se encuentra un clasificador lineal con 3 capas fully-connected con 4096 canales en las dos primeras y 1000 canales en la última (las 1000 clases de ILSVRC) seguido de una capa de softmax.

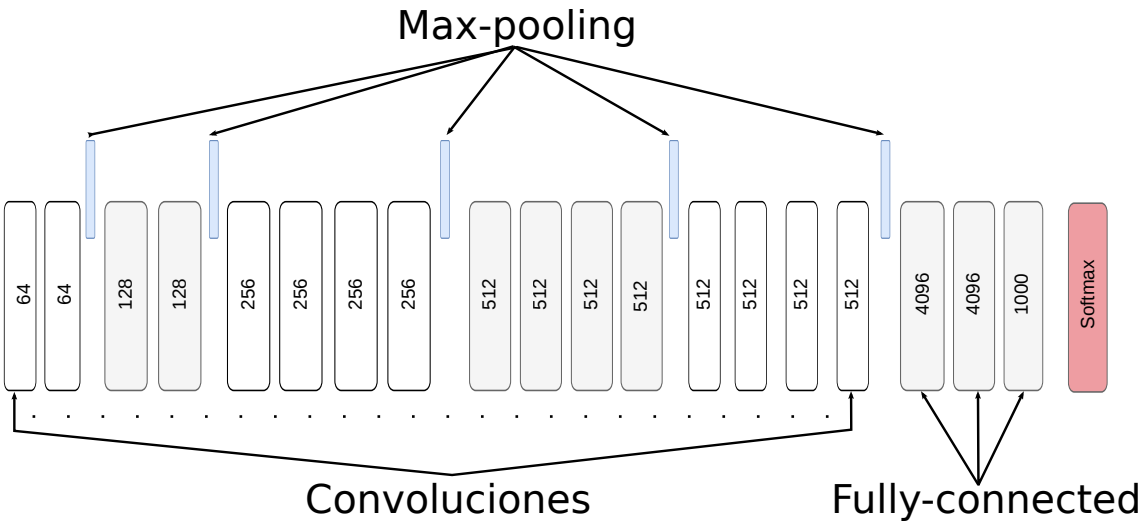


Figura 4.1: Arquitectura completa de la red VGG Net de 19 capas con pesos.

La red fue entrenada mediante la **optimización de la regresión logística multinomial**, con el algoritmo de descenso de gradiente con pequeños lotes (cuya traducción en inglés será *batches*) de imágenes [12]. El tamaño de los lotes es de 256 y el momentum 0,9. Hacen uso de regularización L_2 para la penalización de pesos grandes con factor de 5×10^{-4} y existe dropout [24] en las dos primeras capas fully-connected con probabilidad 0,5 de frenar las activaciones. El coeficiente de aprendizaje (LR) al comienzo del entrenamiento es de 10^{-2} reducido cada vez que el coeficiente de pérdida deja de disminuir.

El entrenamiento les llevó de **2 a 3 semanas** dependiendo de la red, usando 4 GPUs NVIDIA Titan Black. Para su inicialización fueron entrenando gradualmente las redes comenzando con la de 11 capas con pesos (inicializada con muestras de la distribución normal), de manera que una vez esté

entrenada sea posible transferir sus pesos a la de 13 capas de profundidad, siguiendo con este método hasta la red de 19 capas.

4.2. Resultados con clip-arts

La red VGG Net 19 [23] funciona de manera muy precisa con fotografías, obteniendo casi un 93 % de precisión top-5 en el ILSVRC [20]. Sin embargo, uno de los inconvenientes de las redes neuronales profundas son sus pobres resultados a la hora de generalizar un problema, haciendo que no funcione del mismo modo si sus entradas son imágenes de otro tipo. Para realizar pruebas se separan los conjuntos de imágenes de clip-art usando un 80 % para el futuro entrenamiento y validación y un 20 % para probar el modelo. Haciendo uso del conjunto de imágenes de test para evaluar su desempeño se observa como la red obtiene unos porcentajes de error mucho más elevados, como se aprecia en la Tabla 4.1 (entorno a un 60 % más en error top-1, un 70 % más en error top-5, que los obtenidos con ImageNet).

		Test			
Red empleada	Métrica	I	N	N23	C
Arquitectura base	Prec. Top-1	66,10	4,80	13,40	26,50
	Prec. Top-5	86,95	12,20	31,70	47,40

Tabla 4.1: Tabla con precisión top-1 y top-5 haciendo uso de la arquitectura base (VGG Net 19) con todos los conjuntos de datos empleados en el trabajo: ImageNet (I), Noisy (N), Noisy 23 (N23) y Curated (C).

Las características a bajo nivel de las imágenes de clip-art cómo **bordes o colores** e incluso características localizables por la red a medio nivel como las **texturas**, difieren mucho de las que podrían encontrarse en una fotografía. La red, dado su entrenamiento con ImageNet, ha aprendido una representación de las clases donde cada una de ellas tiene unas características de color, una forma y una textura concretas. Por el contrario, los clip-arts no tienen ninguna restricción impuesta y una clase que aparentemente debería ser fácil de clasificar se convierte en algo complejo para la red neuronal, como se ve en la Figura 1.1, dada esa componente abstracta que le puede dar el artista a la imagen que está creando.

Si se analizan los resultados individualmente del conjunto de datos noisy (23 clases) mostrados en

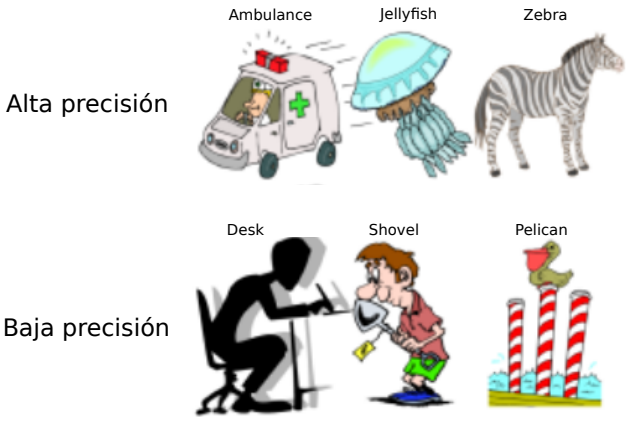


*Figura 4.2: Comparativa de imágenes predichas con mejor precisión por la red (*ambulance, jellyfish *y* zebra*) e imágenes predichas con los peores resultados (*desk, shovel *y* pelican*) con el fin de mostrar las diferencias a la hora de identificar los objetos entre ambas.*

la Figura 4.3 se aprecia como las clases *ambulance*, *jellyfish* y *zebra* tienen un mejor resultado que otras como *pelican*, *desk* o *shovel*. Esto se debe a que las primeras clases poseen en su mayoría imágenes donde el objeto además de ser fácilmente identificable se encuentra centrado. Por el contrario, en las clases con menor precisión y mayor error de máxima entropía los objetos son más complicados de identificar y generalmente son parte de una acción con otro objeto que es el principal. Asimismo, algunas imágenes no representan el objeto de la misma manera que se apreciaría en una fotografía, teniendo pelícanos verdes o rosas, provocando que la red sea incapaz de identificarlos (ver Figura 4.2).

Haciendo la limpieza manual de los datos es posible eliminar problemas de ambigüedad de las imágenes, e incluso, algunos errores con imágenes donde el objeto a clasificar no es identificable. Aun con estas soluciones las características a bajo nivel y medio nivel siguen siendo un inconveniente. Es por ello, que, a pesar de tener un conjunto de datos curado, los resultados siguen siendo inferiores a los obtenidos con ImageNet [5] como se ha visto en la Tabla 4.1, influenciados principalmente por los problemas con las características a bajo y medio nivel de las imágenes que la red es incapaz de capturar en sus capas bajas y medias.

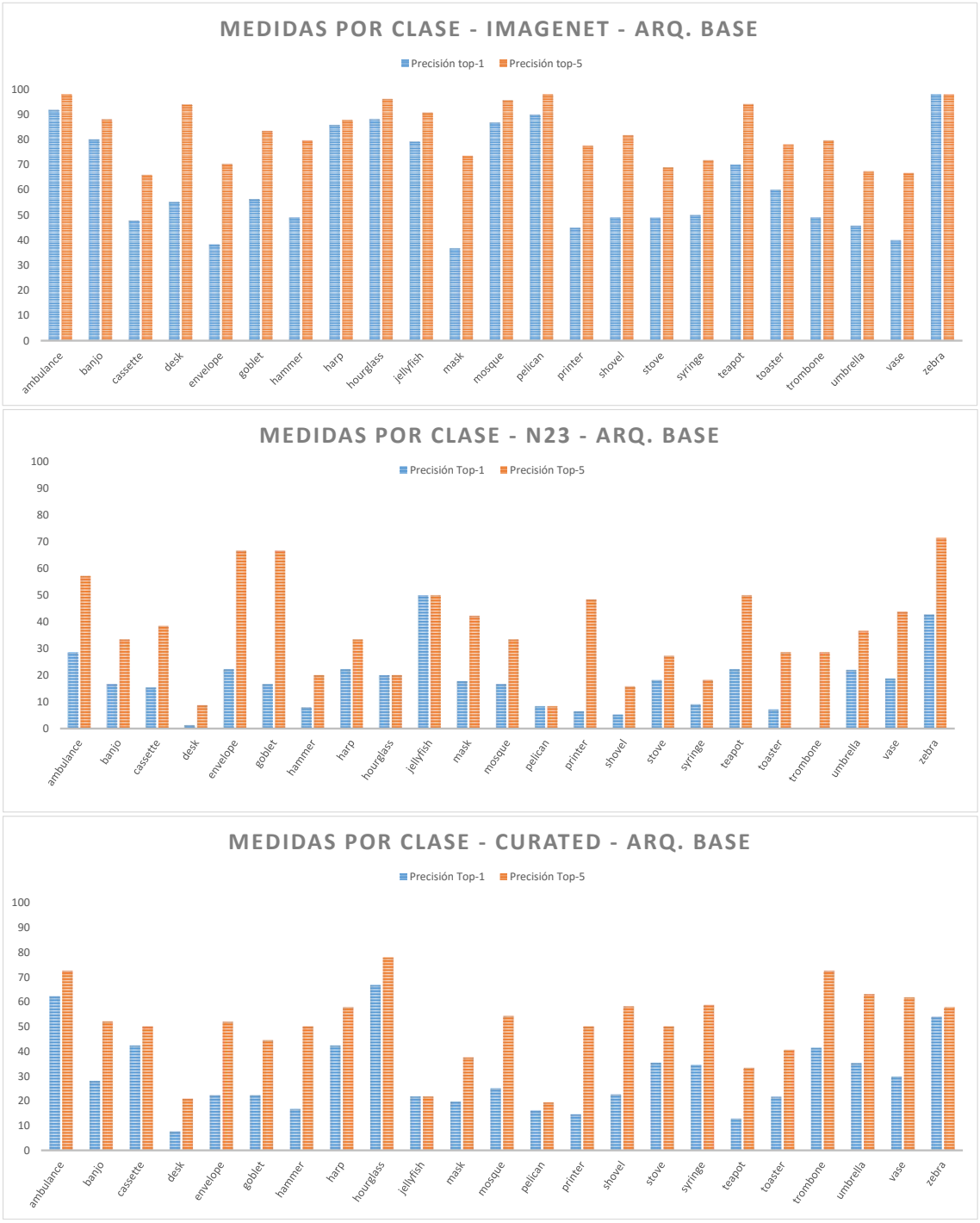


Figura 4.3: Gráfico con los cálculos por clase para los conjuntos Noisy 23, ImageNet y Curated con la red VGG Net 19.

5 ARQUITECTURAS PROPUESTAS

Uno de los problemas de los algoritmos que hacen uso de técnicas de aprendizaje automático es que suelen ser de propósito específico, lo que implica que sólo son capaces de resolver los problemas para los que fueron diseñados sin tener la capacidad de generalizarlo. Por ejemplo, la red usada en este trabajo, VGG Net 19, como ya hemos visto funciona de manera precisa clasificando fotografías, pero, si las imágenes son clip-arts su precisión disminuye drásticamente, en torno a un 70 % en la precisión top-1.

Este capítulo explica en la Sección 5.1 las decisiones tomadas sobre la arquitectura de la red para obtener mejor resultado a la hora de clasificar el conjunto de imágenes de clip-art, en la Sección 5.2 se detalla el proceso de optimización de los híper-parámetros para realizar de manera precisa la tarea de clasificación de imágenes sobre un nuevo conjunto de datos.

5.1. Arquitectura base + SVM

Un tema activo dentro de la investigación en Deep Learning es la **transferencia de conocimiento** [16] [4], las técnicas que permiten a una red neuronal entrenada para un tipo de dato tener porcentajes de precisión elevados sin necesidad de volver a entrenarla con otro tipo de dato diferente. Siguiendo esta línea se propone hacer uso de una SVM que será entrenada con las activaciones que

produce el nuevo conjunto de datos en las capas altas, las que capturan la información a alto nivel [28].

La Figura 5.1 corresponde a una visualización de las activaciones obtenidas de la red VGG Net 19 [23] en su segunda capa fully-connected haciendo uso del algoritmo t-SNE [26]. Dicho algoritmo reduce la dimensionalidad de los vectores con las activaciones (en este caso a dos dimensiones), haciendo posible su visualización en un gráfico. Analizando el gráfico se observa que objetos de la misma clase están agrupados en zonas cercanas, lo que verifica que la red es capaz de extraer descriptores de cada clase a alto nivel. Estos descriptores servirán para el entrenamiento de la SVM que los clasificará en una de las 23 clases semánticas a las que pertenecen las imágenes con el objetivo de sobreponer los problemas vistos en las Secciones 3.2 y 4.2. Esta modificación de la arquitectura tiene 2 cambios principales :

- Primero se eliminan las capas softmax y la última fully-connected (FC), de manera que la salida de la red es ahora la segunda capa FC, que se trata de un vector de características (feature vector) de tamaño 4096.
- A continuación, se hace uso de los feature vector obtenidos para alimentar una máquina de soporte de vectores (SVM) que será primeramente evaluada para obtener los parámetros óptimos, posteriormente entrenada con los datos de entrenamiento del conjunto curated y finalmente usada para la clasificación de los feature vector extraídos de cada imagen.

La Figura 5.2 muestra una visión global de la arquitectura descrita, conteniendo la SVM para la clasificación.

En la Figura 5.3 se puede observar el flujo de trabajo del algoritmo, donde para poder cargar la red en la GPU se emplean las herramientas Torch y Caffe, que permiten obtener las activaciones en cualquier capa de la red que serán guardadas como un fichero HDF5. Para entrenar la SVM se hace uso de Python y scikit-learn, que podrá leer los HDF5 generados con las activaciones extraídas, para, posteriormente, haciendo uso de validación cruzada con el 20 % de los datos usados para entrenamiento, obtener los mejores parámetros de la SVM.

A la hora de entrenar la SVM se hace uso del 80 % de las imágenes del conjunto de datos curated como entradas para el entrenamiento, de este 80 % un 20 % se usa para realizar la **validación cruzada** y obtener los mejores parámetros, mientras que el 20 % restante serán los datos de test. Los mejores

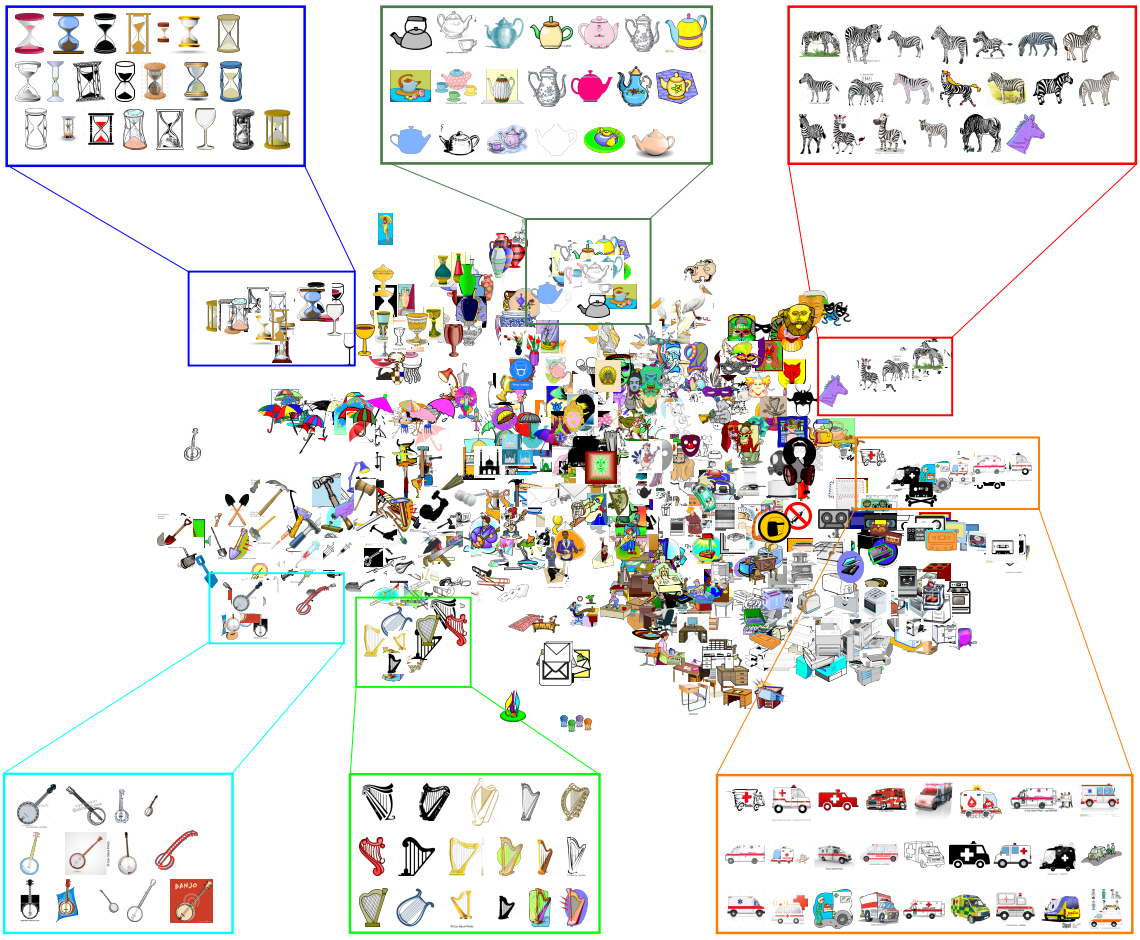


Figura 5.1: Visualización de las activaciones de las capas altas haciendo uso del algoritmo t-SNE [26] sobre los datos de validación del conjunto curado, donde se pueden observar grupos de imágenes próximas que comparten la misma clase.

parámetros obtenidos por las activaciones generadas con la red pre entrenada con ImageNet [5] son el kernel de la **función de base radial** (RBF son sus siglas en inglés), que hace uso de la distancia euclídea al cuadrado para la clasificación, un parámetro $C = 1$ que permite realizar ciertos errores en la clasificación para ganar mayor estabilidad en el modelo (a mayor C menor permisividad en los errores), y por último un valor $\gamma = 0{,}0001$ que marca el peso de un ejemplo en el entrenamiento. Siendo la ecuación del kernel: $\text{RBF} = \exp(-\gamma|x - x'|^2)\gamma$. Además, existe un último parámetro, la función de decisión, que es uno contra el resto (One versus the rest, cuyas siglas en inglés son OVR) que implica entrenar un solo clasificador por clase con los ejemplos de esa clase como positivos y el resto como negativos.

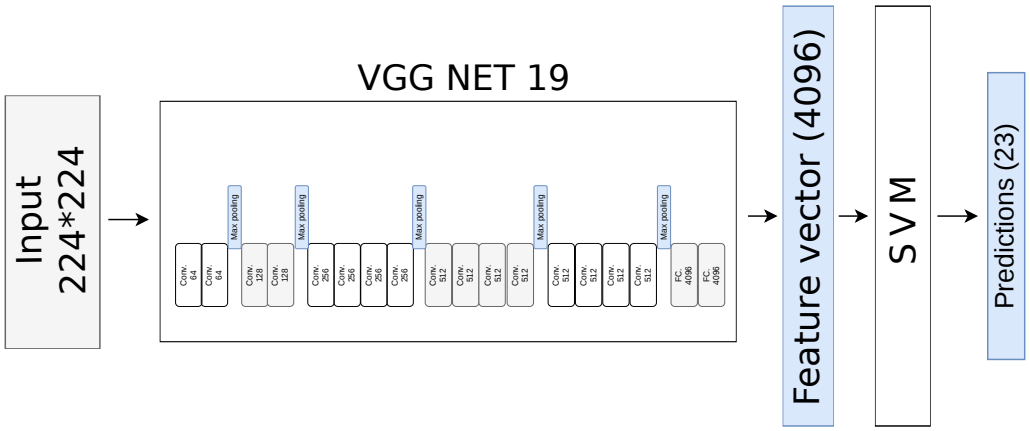


Figura 5.2: Nueva arquitectura de red obtenida tras añadir la SVM.

Algorithm 2 Ejemplo de clasificación con la función OVR

```
procedure OVR(Clasificador C, Etiquetas L, Entradas X)
    //L_i corresponde a la etiqueta de la imagen en el índice i de X, es decir a la imagen X_i//
    for etiqueta l en L do
        Z ← nuevo vector etiquetas
        for i = 1 HASTA sizeof(Z) do
            if l_i = l then
                z_i = 1
            else
                z_i = 0
        pred ← Usar C con (X, Z)
```

El entrenamiento lleva tan solo unos minutos y los resultados obtenidos son prometedores, mejorando los porcentajes de acierto alrededor de un 35 % y disminuyendo el error en la función de máxima entropía en hasta 3,5 puntos, tal y como puede verse en la Tabla 5.2.

Dado el poder de abstracción de la red VGG Net 19 a alto nivel, muchas de las carencias que tenía, explicadas en las Secciones 3.2 y 4.2, son posibles de superar con las activaciones generadas. Estos descriptores clasificados haciendo uso de una SVM devuelven porcentajes de acierto mucho más elevados y un error de máxima entropía más bajo global e individualmente, como se aprecia en la Figura 5.4.

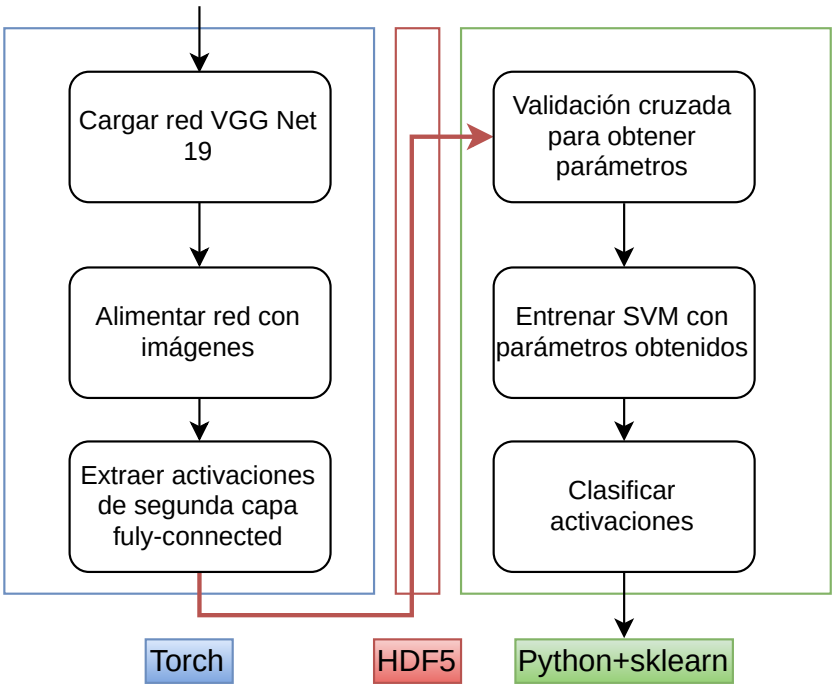


Figura 5.3: Flujo de trabajo del algoritmo para poder clasificar imágenes de tipo clip-art. El primer paso será cargar la red en la memoria de la GPU para, posteriormente, poder alimentarla con las imágenes y generar así las activaciones de la segunda capa fully-connected que serán almacenadas en un fichero con formato HDF5. La aplicación encargada de la SVM leerá el conjunto de activaciones y obtendrá las de validación a partir del conjunto de entrenamiento para realizar la validación cruzada obtener los parámetros óptimos, entrenar la SVM y, finalmente, clasificar las activaciones de test para obtener las predicciones.

5.2. Optimizar la red

Las mejoras obtenidas con la SVM son muy esperanzadoras, pero sin hacer el uso de la máquina de soporte de vectores (SVM) se observa como la red no consigue altos porcentajes de acierto, concluyendo que las características a bajo y medio nivel podrían estar lastrando su desempeño. Para solucionar este problema, se va a reentrenar la red para buscar unos pesos (W) y bias (b) en cada capa que entiendan de mejor forma los nuevos datos de entrada sin perder su poder de representación a alto nivel. Para el entrenamiento se hace uso del conjunto de imágenes **noisy**, dada la cantidad de datos que necesitan las redes neuronales profundas. A pesar de los problemas que se han visto: posibilidad

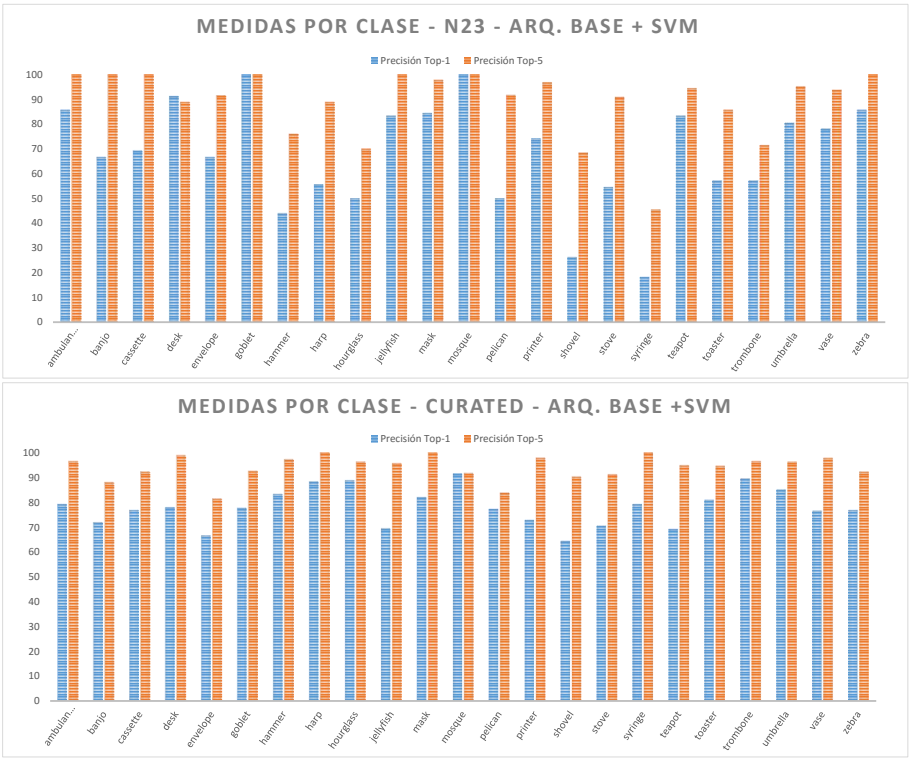


Figura 5.4: Resultados obtenidos por la red con uso de softmax y una SVM que clasifica las activaciones obtenidas de la segunda capa fully-connected.

de representaciones abstractas, diferentes texturas, colores y bordes, y la diferente representación de un mismo objeto con el paso del tiempo; este conjunto es de gran utilidad ya que el objetivo principal es modificar ligeramente (no realizar un reentreno completo, para el cual no sería útil) la arquitectura para que capture las características a bajo y medio nivel que antes le era imposible, y genere un descriptor de la imagen a alto nivel con mayor precisión.

5.2.1. Modelos exploratorios sin resultado

Con el objetivo de **optimizar la red** para los nuevos datos de entrada se reentrenará partiendo de los pesos (W) y bias (b) que ha obtenido tras su entrenamiento en ImageNet. Para poder reentrenar la red VGG Net 19 se modifica la última capa softmax, sustituyéndola por $\log(softmax)$ a la que se aplica la función de pérdida *negative-log-likelihood* que, actuando conjuntamente dan como resultado el error de máxima entropía. Algunos de los modelos reentrenados hacen uso de una nueva capa que aplica una función llamada normalización de lotes (cuya traducción es *batch-normalization*) [9] que centra cada activación de los lotes en media cero y varianza unitaria, dicha media y varianza son medidas independientemente en cada activación generando una compensación y un coeficiente de multiplicación que serán aplicados a las activaciones. Esta ligera modificación hace que las arquitecturas profundas aprendan más rápido y sean más estables.

Dado que la optimización de los parámetros es un proceso costoso se tuvieron que probar un conjunto de modelos que buscaban solucionar diferentes problemas para el reentrenamiento. A continuación, se explican todos los modelos probados que no tuvieron un resultado satisfactorio pero que gracias a sus resultados se pudo realizar una configuración de las capas que fuera óptima para el problema de la clasificación de imágenes de clip-art:

(A) Reentreno como en ImageNet El primer modelo probado sigue el entrenamiento realizado sobre ImageNet. Se hace uso de un tamaño de lotes de 8, un momentum establecido a 0.9, coeficiente de aprendizaje $1e-3$ con caída en $5e-4$ y cada epoch llevará a cabo 10,000 pasos del algoritmo SGD. Tras realizar 15 epochs se observa que la función de pérdida no ha reducido su valor medio, indicando que los gradientes no son estables y hacen zigzaguear al algoritmo que no es capaz de encontrar el camino hacia el mínimo global.

(B) Uso de normalización de lotes Dado el resultado insatisfactorio del modelo A, se prueba a

incluir la capa de normalización de lotes entre cada capa convolucional y se mantienen todos los parámetros tal y como estaban anteriormente. Se puede observar como el nuevo modelo entrena con el conjunto de datos de entrenamiento, pero el coeficiente de pérdida en el conjunto de test no reduce, concluyendo que a pesar de la mejora con la normalización de lotes entre convoluciones, esta no es la inicialización correcta, pues estaba incurriendo en overfitting con los datos de entrenamiento.

(C) Bloqueando backpropagation en capas altas Visto que el entrenamiento manteniendo un coeficiente de aprendizaje igual en todas capas, el usado en el modelo B, no ha dado resultados satisfactorios, se comienza a explorar nuevas vías relativas a la investigación de Zeiler y Fergus [28]. Se conoce que las capas bajas y medias no responden bien en la red VGG Net 19, dadas las diferencias en las entradas usadas para entrenar y las nuevas imágenes, se elimina la actualización del algoritmo SGD en las capas altas (últimas 8 convoluciones) evitando que sus pesos y bias se actualicen, para así poder conservar su capacidad de representación a alto nivel. Se mantienen los parámetros iguales que en los modelos anteriores. El resultado final vuelve a ser insatisfactorio, la función de pérdida logra descender hasta estancarse en $6,8$. Confiando que quizás se habían restringido demasiado las capas medias se repite el mismo modelo eliminando la actualización de los gradientes, los pesos y el bias solo en las 6 últimas convoluciones con el mismo resultado negativo, la función de pérdida deja de reducir alrededor de $6,8$.

(D) Reinicio de pesos en capas bajas Los resultados evitando que el modelo aprenda en las capas altas no funcionan como era esperado (modelo C), por ello, se comienza a creer que los pesos que la red ha aprendido a bajo nivel son ineficaces para este problema, se decide reiniciarlos (en las 6 primeras capas) cogiendo muestras de la distribución uniforme y aplicarles un coeficiente de aprendizaje de $1e-3$. Las capas altas (las 6 últimas) volverán a tener su actualización de pesos y bias bloqueada, y el resto de la red neuronal se actualiza con un coeficiente de aprendizaje menor: $1e-4$ evitando perder la capacidad de abstracción a alto nivel o que el clasificador lineal (las capas fully-connected) empeore lo aprendido. En este caso el modelo deja de aprender, cada epoch el coeficiente de pérdida es mayor, concluyendo que, evidentemente, esta inicialización no es correcta. Se decide aplicar el mismo modelo aumentando el coeficiente de aprendizaje en las capas bajas a $1e-2$, confiando en que los gradientes sean menos inestables y el algoritmo

SGD sea capaz de avanzar hacia el mínimo global. El resultado final será una leve mejora del modelo en los 5 primeros epochs, que acaba concluyendo con su estancamiento, haciendo que a efectos prácticos el modelo no haya variado sus porcentajes de error y precisión.

(E) Reinicio del clasificador lineal Visto que ninguna de las aproximaciones al problema anteriores ha dado un resultado satisfactorio, se intenta explorar una solución alrededor del clasificador lineal. Se espera que este haya aprendido unos pesos y bias condicionados a las imágenes naturales de la base de datos ImageNet que podrían estar lastrando el proceso de aprendizaje de la red globalmente. Se reinician sus pesos y bias, cogiendo muestras de la distribución uniforme. De nuevo, se bloquea la propagación de pesos y bias en las 6 últimas capas con el fin de conservar el poder de abstracción a alto nivel que tiene la red VGG Net 19. Además, se modifica el coeficiente de aprendizaje del clasificador lineal para que este aprenda más rápido a $1e-2$ y evitar así cierta inestabilidad que podría ser causada en los gradientes. El modelo comienza a aprender, pero tras 10 epochs, se vuelve inestable, no reduciendo su coeficiente de pérdida y, por tanto, no siendo una solución al problema. También se prueba a eliminar la propagación de pesos y bias, para verificar que la aproximación al problema no es incorrecta, pero se obtiene mismo resultado, gradientes inestables tras cierto número de epochs y el modelo sin entrenar.

5.2.2. Modelo óptimo

Finalmente, se intenta combinar las técnicas usadas anteriormente en una con el fin de probar si el modelo, reiniciando sus capas medias y bajas (modelo D), y su clasificador lineal (modelo E), es capaz de abstraerse lo suficiente como para mejorar con imágenes de clip-art. Las capas bajas y medias (desde la primera convolución hasta la décima) son reiniciadas inicializando sus pesos y bias con muestras de la distribución uniforme, además su coeficiente de aprendizaje es establecido en $1e-2$ con el fin de que los gradientes se estabilicen y aprendan lo más rápido posible (la caída de LR hará que reduzca si deja de aprender). En las capas altas el coeficiente de aprendizaje se reduce a $1e-4$ evitando así que se modifiquen de manera acusada (manteniendo su capacidad de descripción a alto nivel) pero permitiendo cierta adaptación a los nuevos datos. El clasificador lineal es reiniciado igual que las capas bajas y su coeficiente de aprendizaje será establecido a $1e-2$ nuevamente, evitando que se haya adecuado en exceso a las imágenes de ImageNet y permitiendo que aprenda de nuevo

como clasificar clip-arts y genere activaciones teniendo en cuenta las características de bajo y medio nivel que ha de aprender. Puede verse un ejemplo de los cambios en el modelo para el entrenamiento en la Figura 5.5.

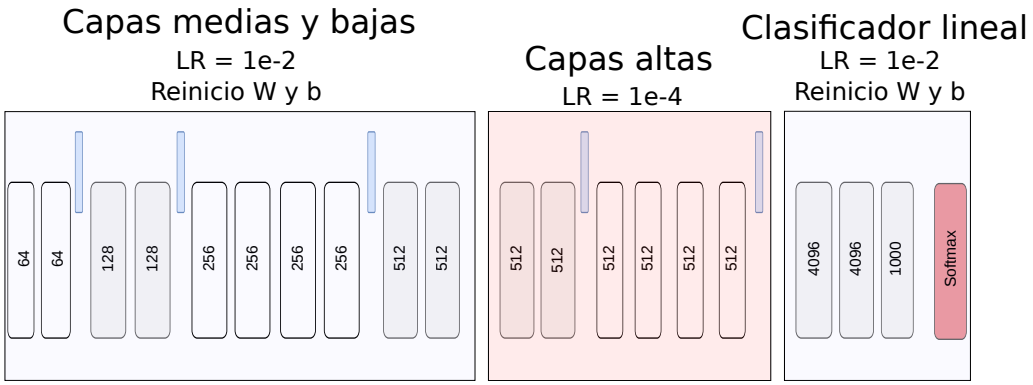


Figura 5.5: Arquitectura con los cambios aplicados en las diferentes capas para el entrenamiento. Los fondos azules marcan las capas cuyo coeficiente de aprendizaje se ha establecido en $1e-2$ *(para aprender más rápido) y cuyos pesos se han reiniciado usando muestras de la distribución uniforme, mientras que el fondo rojo denota un coeficiente de aprendizaje más reducido* $1e-4$ *(evitando su modificación).*

Detalles del entrenamiento

El entrenamiento comienza reduciendo la función de pérdida tanto en el conjunto de entrenamiento como en el de test, lo que es un buen indicio. Tras 5 días de entrenamiento y 100 epochs la precisión top-1 en el conjunto de datos **Noisy 23** ha aumentado en un 5 % en el conjunto de datos de test hasta llegar a 17,65 % y las medidas de la función de pérdida hacen pensar que aún tiene un largo recorrido hasta converger. A pesar de mejorar globalmente, el hecho de realizar el entrenamiento sobre el conjunto de datos noisy, crea un problema visible en las gráficas de resultados (ver Figura 5.6) la red modifica sus pesos y bias en las capas bajas teniendo una mayor influencia de las clases donde hay un mayor número de entradas, haciendo que las clases con menor número no mejoren o incluso empeoren sus resultados. Las clases como *ambulance* con escasos 20 ejemplos de entrenamiento (ver Tabla 3.1) son incapaces de influir en la red neuronal profunda, que necesita de una gran cantidad de datos para poder entrenar correctamente. Además el hecho de reiniciar las capas bajas y medias de la red podría estar causando problemas relativos al olvido de lo previamente

aprendido con el entrenamiento sobre ImageNet. Al realizar el reinicio, la red podría estar perdiendo el poder de abstracción adquirido para algunas de las clases provocando que su precisión empeore.

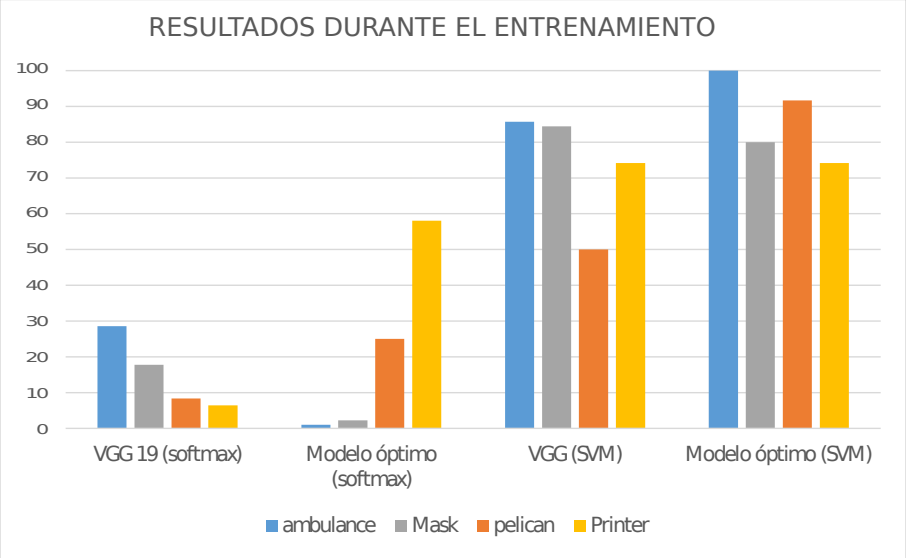


Figura 5.6: Visualización de la perdida de precisión al optimizar los parámetros de la red para los nuevos datos.

Tras otros 5 días de entreno se alcanza el epoch 200, la precisión del nuevo modelo sigue aumentando, mientras que las clases sin representación suficiente continúan con peores resultados. El modelo alcanza ahora un 21,18 % de precisión top-1 global, y con cierto margen de mejora dado que la función de pérdida sigue reduciendo, aunque en menor medida (ver Figura 5.7).

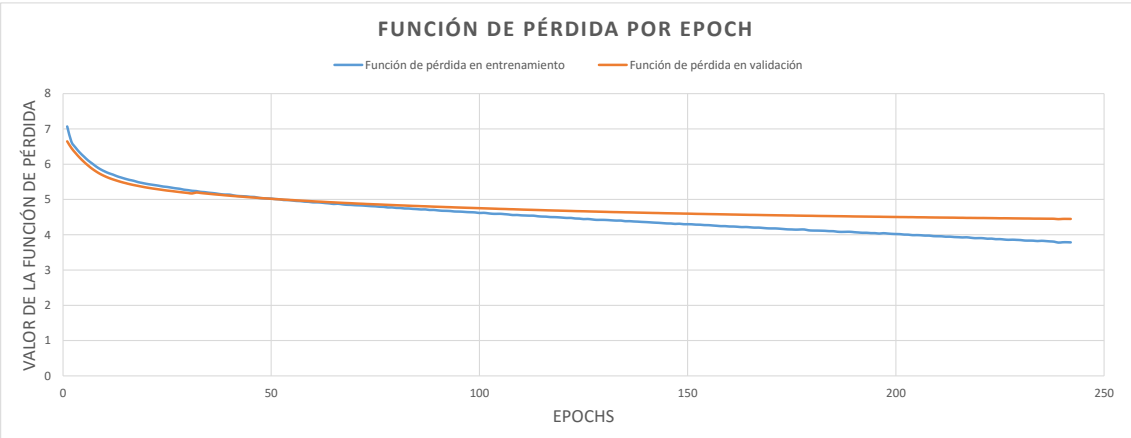


Figura 5.7: Valores tomados por la función de pérdida durante el entrenamiento.

Finalmente, en el **epoch** 239, tal y como se observa en la Figura 5.7 se para el entrenamiento tras no mejorar durante 3 iteraciones seguidas (usando la técnica de parada temprana para evitar overfitting). El entrenamiento llevo **12 días** empleando una gráfica NVIDIA GTX980 Ti, obteniendo un 21,88 % de precisión top-1 global, mejorando por tanto la precisión top-1 de la red VGG Net 19 [23] sin optimización en un 8 %. La Figura 5.8 muestra los resultados por clase en los conjuntos noisy y curated, la red es capaz de aumentar su precisión tanto top-1 como top-5 de manera global, lo que podría ser un indicio de que los problemas relativos a las imágenes artísticas (la abstracción, las características a medio y bajo nivel o el hecho de que el etiquetado de una imagen no sea una parte principal de ella) puedan ser parcialmente sobrepuestos realizando una optimizando de los híper-parámetros.

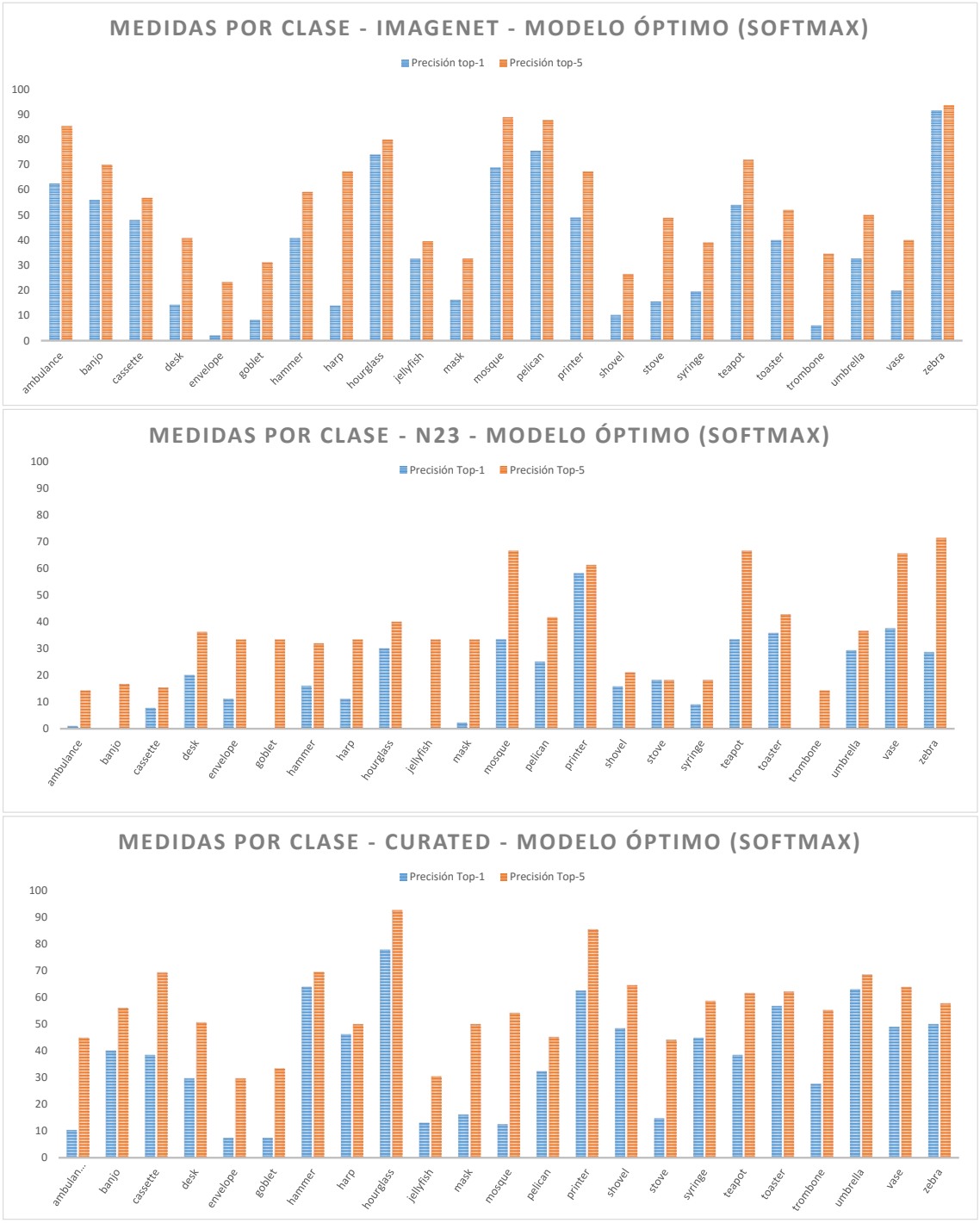


Figura 5.8: Resultados tras el entrenamiento para los conjuntos de datos ImageNet, Noisy 23 y Curated, mostrando cómo las clases con menor número de ejemplos, dada la actualización en pesos y bias de la red, tienen un peor resultado.

Como se ha mencionado, a la hora de entrenar, podrían existir problemas relativos al número de entradas por clase o al hecho de olvidar lo previamente aprendido con ImageNet. Esto provoca que algunas clases puntuales obtengan peores resultados de precisión mientras que el resto mejore, haciendo que el error global de la red disminuya, como se aprecia en la Tabla 5.2.

El problema con las clases particulares que disminuyen su precisión no será de gran relevancia, pues a nivel global, los resultados son mejores y por tanto, la red está interpretando de mejor forma las características que antes era incapaz de reconocer. Tras **extraer las activaciones** de la segunda capa fully-connected de los datos de entrenamiento del conjunto de imágenes curated se vuelve a realizar validación cruzada sobre los parámetros del SVM, obteniendo en este caso el kernel sigmoidal ($\tanh(\gamma\langle x, x'\rangle + r)$) y un $C = 10$ manteniendo el resto de parámetros iguales.

		Test	
Red empleada	Métrica	N23	C
Modelo óptimo (con SVM)	Prec. Top-1	82,82	85,89
	Prec. Top-5	95,53	97,10

Tabla 5.1: Resultados globales obtenidos en todos los conjuntos de imágenes tras la optimización de los parámetros de la red, usando una máquina de soporte de vectores para la clasificación.

El entrenamiento, de nuevo, lleva escasos minutos y como se aprecia en la Tabla 5.1 los resultados obtenidos mejoran alrededor de un 7 %, obteniendo una precisión top-1 con el conjunto de datos de test de 85, 89 %, indicando que las activaciones generadas son mejores descriptores de la clase, gracias a la información obtenida a bajo y medio nivel. La Figura 5.9 muestra los resultados obtenidos por clase. En el Anexo D se pueden encontrar más gráficas con resultados obtenidos durante el entreno de la red.

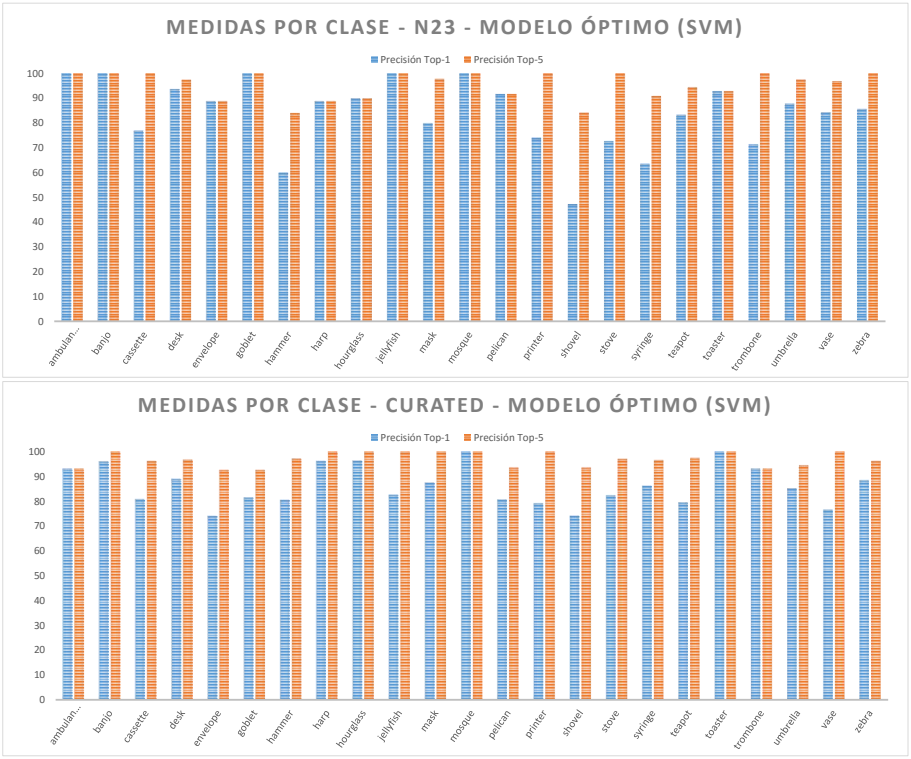


Figura 5.9: La Figura muestran los resultados finales obtenidos tras clasificar el conjunto de test de los conjuntos curated haciendo uso de la SVM.

Resultados finales obtenidos

A continuación, se muestra la Tabla 5.2 que contiene los resultados de la precisión top-1 medidos sobre la arquitectura óptima y la arquitectura base, además de todos los modelos insatisfactorios probados con los motivos por los que se desecharon. En ella se aprecia cómo, el hecho de modificar las capas a bajo nivel y alto nivel hace que la red pierda la capacidad de abstracción e interpretación que antes tenía por un mejor resultado sobre imágenes de tipo clip-art. También contiene los resultados que se obtienen globalmente tras sustituir las dos últimas capas de la red neuronal VGG Net 19 pre-entrenada con ImageNet, así como, tras la optimización realizada por una máquina de soporte de vectores (SVM). Los conjuntos de imágenes empleados para las mediciones son Noisy 23 y Curated dado que la SVM, tras su entrenamiento, solo clasifica esas 23 clases.

La Figura 5.10 muestra las probabilidades obtenidas con la red VGG Net 19 [23] y el clasificador softmax frente al modelo óptimo desarrollado con la SVM entrenada. Además, entre paréntesis,

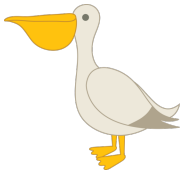
		Test			
Modelo	Métrica	I	N	N23	C
Arquitectura base	Prec. Top-1	66,10	4,80	13,40	26,50
	Prec. Top-5	86,95	12,20	31,70	47,40
A	-	Función de error no reduce			
B	-	Overfitting, Función de error en test no reduce			
C	-	Función de perdida estancada en 6,8			
D	-	Función de perdida estancada de nuevo			
E	-	Aprende durante 10 Epochs luego se estanca			
Modelo óptimo (con softmax)	Prec. Top-1	41,66	11,59	21,80	37,96
	Prec. Top-5	65,72	33,51	39,10	57,30
Arquitectura base +SVM	Prec. Top-1	-	-	72,24	78,22
	Prec. Top-5	-	-	91,53	95,01
Modelo óptimo (con SVM)	Prec. Top-1	-	-	82,82	85,89
	Prec. Top-5	-	-	95,53	97,10

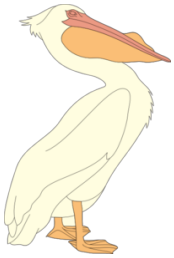
Tabla 5.2: Resultados globales de las distintas arquitecturas para los 4 conjuntos de imágenes empleados: Imagenet (I), Noisy (N), Noisy (23), Curated (C).

se encuentra la posición que ocupa la probabilidad de que sea la clase *pelican*, ordenando todas las probabilidades de cada clase en orden descendente. Puede apreciarse una amplia mejora en las imágenes a las que la red daba una probabilidad realmente baja. Por otro lado, las imágenes que tenían una probabilidad alta la tienen ahora algo mayor, confirmando que los cambios producidos en la arquitectura hacen que las clasificaciones sean ahora más robustas.

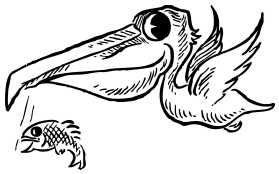



Modelo óptimo + SVM	0.87771 (1)	0.63657 (1)	0.99746 (1)	0.99169 (1)
VGG19 + softmax	0.00009 (462)	0.000007(634)	0.99436 (1)	0.01037 (19)






Modelo óptimo + SVM	0.93697 (1)	0.97398 (1)	0.86552 (1)	0.98490 (1)
VGG19 + softmax	0.000002 (708)	0.00922 (15)	0.01262(11)	0.00358 (51)


Modelo óptimo + SVM	0.98055 (1)	0.51279 (1)	0.64120 (1)	0.99109 (1)
VGG19 + softmax	0.07654 (2)	0.00004 (582)	0.00003 (541)	0.00021 (147)

Figura 5.10: Comparativa de probabilidades obtenidas por la red VGG Net 19 + softmax y el modelo óptimo + SVM para un subconjunto de imágenes de la clase pelican. También se muestra la posición que ocupa dicha probabilidad en el conjunto de probabilidades de pertenecer a cada clase devueltas por ambas arquitecturas.

6 CONCLUSIÓN Y TRABAJO FUTURO

En este proyecto hemos explorado un conjunto de técnicas del aprendizaje automático, conocidas como Deep Learning, que consisten en incluir capas no lineales en las redes neuronales tradicionales, aumentando su profundidad, capacidad de abstracción y de síntesis, para lograr una solución al problema de la clasificación de imágenes artísticas o ilustraciones.

Durante su desarrollo se observa como un problema relativamente sencillo para el ser humano, se convierte en una tarea realmente compleja de realizar por un ordenador. Gracias a la comprensión completa de la forma de trabajo de las redes neuronales convolucionales, su arquitectura y los algoritmos de optimización usados, es posible encontrar una solución al problema, que, sin ser perfecta, devuelve unos resultados realmente esperanzadores.

Por tanto, mostramos cómo las redes neuronales convolucionales profundas pueden ser optimizadas haciendo que, en una tarea muy compleja para los ordenadores, como es la clasificación de imágenes artísticas o ilustraciones, se consigan porcentajes de error realmente bajos gracias al uso de las activaciones obtenidas por la red a alto nivel su clasificación con una máquina de soporte de vectores (SVM) y la optimización de las capas bajas de manera que, la red sea capaz de interpretar imágenes con texturas, colores y bordes que no corresponden con los de una imagen natural.

6.1. Gestión del proyecto

El proyecto comenzó a realizarse a principios de febrero del año 2016, el desarrollo de este se realizó de manera presencial durante los meses lectivos y de manera remota el resto del tiempo. Desde el comienzo se acordó trabajar alrededor de 4h horas diarias y realizar reuniones semanales con el fin de llevar un control del avance.

1. La primera tarea acordada fue un estudio del campo de la clasificación de imágenes y la creación arte con redes neuronales profundas, con el fin de comprender las últimas técnicas usadas así como sus algoritmos.

2. Conocido el trabajo previo en clasificación de imagen y creación de arte, su desempeño y forma de trabajo, se discutió una propuesta de solución con Elena (directora del proyecto) que fue presentada como un proyecto del Graphics and Imaging Lab en una de sus reuniones semanales, obteniendo nuevos comentarios, sugerencias e ideas.

3. Conocida la línea a seguir, se comenzó con la obtención y procesamiento de los datos y la decisión de una red neuronal convolucional profunda a usar. Con ambos, se pudieron adaptar los datos para que fueran válidos y usables en la red.

4. Posteriormente, se desarrollaron las pruebas para evaluar el desempeño de cualquier red con nuevos datos como las imágenes de clip-art.

5. Al ver que sus porcentajes de error eran muy bajos se comenzó a desarrollar el código para el entrenamiento y la evaluación de la SVM.

6. Finalmente, viendo los resultados prometedores, y la posibilidad de mejorarlos, se desarrolló la optimización y reentreno de la red.

7. Además, se programó el código para entrenar un auto codificador para usarlo en la clasificación de imágenes de clip-art. Dado el tiempo consumido por los reéntrenos sucesivos no pudo llegar a probarse.

En la Tabla 6.1 puede verse el tiempo empleado en cada tarea de las explicadas anteriormente, asimismo, en la Tabla 6.2 hay un horario detallado con el tiempo en horas invertido en cada tarea.

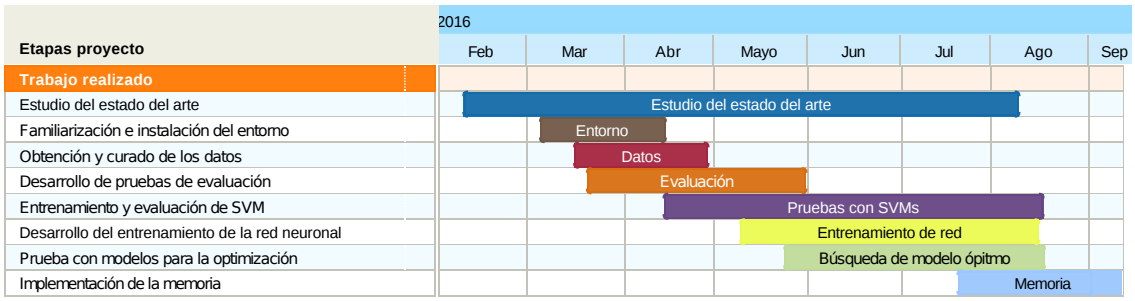


Tabla 6.1: Diagrama mostrando el tiempo que se ha estado realizando cada tarea durante la realización del proyecto.

Tarea	Horas personales	Horas de máquina
Estudio	102	-
Adaptación/instalación framework	37	5
Obtención de datos	46	12
Pruebas de evaluación	37	19
Entrenamiento y evaluación SVM	47	25
Código para el entrenamiento	31	-
Reentrenamiento nuevos modelos	63	350
Implementación de la memoria	115	-
Total	478	411

Tabla 6.2: Tabla con las horas invertidas en cada tarea y el total de estas.

6.2. Trabajo futuro

En la realización del proyecto además de explorar y estudiar técnicas que ayuden a atajar un problema en cuestión, también aprendimos a visualizar posibles caminos a seguir que podrían dar un resultado mejor del esperado. Además, durante el trabajo han aparecido nuevas investigaciones inspiradoras que pueden marcar nuevas líneas a considerar.

Una nueva solución al problema viene directamente de la arquitectura usada y los nuevos parámetros a optimizar a la hora de entrenar la red con coeficientes de aprendizaje (LR) selectivos en cada capa.

- Búsqueda de la configuración optimiza en cada LR de cada capa. Una aproximación sería considerar un número de epochs pequeño probando un conjunto de valores del LR para cada capa (similar a la técnica usada para buscar los valores óptimos de la máquina de soporte de vectores, explicado en la Sección 5.1).

- La segunda opción sería averiguar qué capas necesitan un reinicio de los pesos e inicialización con la distribución uniforme. De la misma manera que con la búsqueda del LR óptimo se podría programar un reinicio de la capa en cuestión.

El hecho de hacer uso de reinicios y la obtención de muestras de distribuciones estadísticas, ofrece una nueva posibilidad. Se ha demostrado como los auto codificadores (cuya traducción al inglés es autoencoders) [2] son capaces de generar pesos para la inicialización de redes neuronales de manera precisa. Por ello, el uso de un auto codificador como red para poder trasladar los pesos de las capas reiniciadas sería otra posible futura investigación.

Actualmente, las redes neuronales residuales (Residual neural network, RNN)[1] están obteniendo resultados en el estado del arte en datasets con imágenes de tamaño pequeño como CIFAR [10], dado que estos algoritmos son de caja negra, no se puede predecir si sería capaz de abstraerse de mejor manera que la red VGG Net 19 [23], por ello sería una posible línea a explorar.

Además de las redes neuronales para clasificación de imágenes hay otro tipo de redes con el fin de generar contenido perdido de una imagen, o recolorizar imágenes en blanco y negro. Las adversarial neural networks [8] son capaces de modificar imágenes a nivel de píxel de manera que no se aprecie prácticamente su modificación, pero a la hora de hacer uso de una red neuronal profunda para clasificación sus predicciones cambien completamente. Podría hacerse uso de una adversarial neural network de manera que se modifique ligeramente la imagen de entrada antes de esta ser clasificada haciendo que los resultados mejoren con las predicciones de la red neuronal convolucional.

[1] Observar https://github.com/szagoruyko/wide-residual-networks

6.3. Comentario personal

Este proyecto ha sido mi primera experiencia y contacto con el mundo de la investigación, y, la verdad, es que ha sido algo realmente positivo y gratificante. El poder compartir charlas, comentarios y opiniones con gente que realmente vive y le apasiona lo que hace te llena de satisfacción, teniendo la impresión de que no pude tomar una decisión más acertada cuando Elena me dio la oportunidad de comenzar con este proyecto. Todos los integrantes del grupo, desde Diego hasta los *proyectandos*, son gente fantástica y llena de entusiasmo. Además, el buen ambiente que se respira en el laboratorio, las oportunidades que brinda el grupo, la idea de crear y colaborar en proyectos de repercusión internacional y la pasión por la investigación que emana el GILab han hecho que me decida a continuar por la vía académica, con el plan de realizar el doctorado en un futuro muy cercano.

Bibliografía

[1] Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. Neural codes for image retrieval, 2014.

[2] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 153–160.

[3] Collobert, R., Kavukcuoglu, K., and Farabet, C. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop* (2011).

[4] Crowley, E. J., and Zisserman, A. In search of art. In *Workshop on Computer Vision for Art Analysis, ECCV* (2014).

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09* (2009).

[6] Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision 111*, 1 (Jan. 2015), 98–136.

[7] Garces, E., Agarwala, A., gutierrez, D., and Hertzmann, A. A similarity measure for illustration style. *ACM Transactions on Graphics (SIGGRAPH 2014) 33*, 4 (2014).

[8] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

[9] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167* (2015).

[10] Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research).

[11] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[12] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (Winter 1989), 541–551.

[13] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing* (2001), IEEE Press, pp. 306–351.

[14] Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., and Yu, K. Large-scale image classification: Fast feature extraction and svm training. 2011.

[15] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010), J. Fürnkranz and T. Joachims, Eds., Omnipress, pp. 807–814.

[16] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR* (2014).

[17] Pearson, K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2* (1901), 559–572.

[18] Perronnin, F., Sánchez, J., and Mensink, T. Improving the fisher kernel for large-scale image classification. In *Proceedings of the 11th European Conference on Computer Vision: Part IV* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 143–156.

[19] Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition, 2014.

[20] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[21] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR abs/1312.6229* (2013).

[22] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature 529* (2016), 484–503.

[23] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014).

[24] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15* (2014), 1929–1958.

[25] Strother H. Walker, D. B. D. Estimation of the probability of an event as a function of several independent variables. *Biometrika 54*, 1/2 (1967), 167–179.

[26] van der Maaten, L., and Hinton, G. E. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[27] Zagoruyko, S., and Komodakis, N. Wide residual networks. *CoRR abs/1605.07146* (2016).

[28] Zeiler, M. D., and Fergus, R. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I* (2014), pp. 818–833.

A ¿QUÉ ES DEEP LEARNING?

El aprendizaje profundo o **Deep Learning** son un conjunto de técnicas que consisten en añadir a las arquitecturas de redes neuronales tradicionales capas no lineales aumentado su profundidad y capacidad de abstracción a alto nivel consiguiendo, así, mejores resultados. Estas nuevas técnicas han permitido que los ordenadores realicen tareas que hasta ahora eran solo posibles de hacer (con altos niveles de precisión) por los seres humanos.

El gran número de capas ocultas de este tipo de redes podría suponer un gran problema años atrás dado que la capacidad de cálculo y memoria de las máquinas entonces no era la suficiente para que una tarea de este tipo pudiera realizarse en un tiempo razonable. Con el aumento de la potencia en las GPUs y la aparición de arquitecturas muy prometedoras, el Deep Learning o aprendizaje profundo comenzó a popularizarse hasta que, hoy en día, el estado del arte en campos como la clasificación de imágenes, el reconocimiento del lenguaje humano o la segmentación de imágenes lo componen algoritmos que hacen uso de estas arquitecturas de red.

El siguiente capítulo explica los conceptos básicos necesarios para la comprensión del documento. El Anexo A.1 expone la teoría básica del aprendizaje automático, comenzando con las arquitecturas más sencillas, su funcionamiento y cómo entrenarlas. El Anexo A.2 introduce las capas no lineales a las arquitecturas vistas, y con ello, las redes neuronales profundas (Deep Learning), finalmente, el Anexo A.3 desarrolla la idea de convolución aplicándola a las arquitecturas profundas para obtener

redes convolucionales, que se enfocarán a la resolución de tareas de clasificación de imágenes.

A.1. Conceptos básicos

A continuación, se exponen los conceptos básicos necesarios para realizar predicciones haciendo uso del aprendizaje automático. Durante esta introducción va a crearse una arquitectura sencilla que hará uso del modelo logístico, una función de máxima entropía y la capa softmax, a dicha arquitectura se le van a añadir nuevas capas y funciones que mejoren sus resultados hasta llegar finalmente a un modelo de red neuronal convolucional profunda enfocado a la clasificación de imágenes.

A.1.1. Clasificador logístico

Un clasificador logístico hace uso de la **regresión logística** para clasificar una entrada, por ejemplo, dada una imagen de entrada X aplicando una función lineal ($W \times X + b$) y una función logística (softmax) se obtiene una probabilidad Y donde, la Y_i más cercana a 1 deberá ser la clase real de dicha imagen. También puede verse como un perceptrón multicapa (MLP) cuya entrada ha sido modificada usando una transformación no lineal que la proyecta en un espacio linealmente separable. La regresión logística fue desarrollada por el estadista David Cox [25] en 1958. El clasificador logístico es también llamado clasificador lineal dado que hace uso de una función lineal para obtener sus predicciones. Esta función lineal se trata de una multiplicación y suma de matrices.

$$W \times X + b = Y \tag{A.1}$$

Donde X es la imagen de entrada (el perro, el gato o el pez), W será la matriz de pesos, b el término de bias e Y serán los resultados de aplicar la función lineal a X, también llamados **logits**, cuya representación gráfica puede verse en la Figura A.1. El aprendizaje automático busca optimizar los valores de W y b para que las predicciones sean lo más certeras posibles.

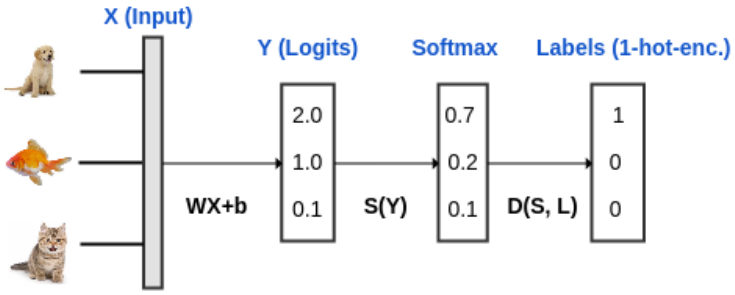


Figura A.1: Ejemplo de arquitectura de red neuronal sencilla con 3 entradas.

A.1.2. Softmax

La función lineal dará como resultado un valor en los números reales, mientras que lo que se necesita es una probabilidad que sea determinante para poder definir a qué clase pertenece la imagen de entrada X, para ello se hace uso de la función **softmax**, S.

$$S(Y_i) = \frac{e^{Y_i}}{\sum_j^n e^{Y_j}} \tag{A.2}$$

El resultado será un conjunto de probabilidades obtenidas a partir de los valores iniciales dados por la función lineal, estas probabilidades serán mayores cuando el valor sea comparativamente grande y pequeños en el caso opuesto, el resultado de aplicar esta función puede verse en la Figura A.1. Softmax mapea un valor a una probabilidad, por lo tanto, la matriz o vector de entrada tendrá el mismo tamaño que el de salida y su suma tendrá resultado 1.

A.1.3. Cross-entropy error

Para poder ver cómo actúa el modelo, y medir su precisión se hace uso de funciones de pérdida, en este caso, la **función de máxima entropía** (D) o *cross-entropy* cuyo resultado será un número en los reales que modela la relación entre dos vectores (vector de predicciones y vector de etiquetas). Dado que se hace uso de logaritmos, cuanto menor sea el resultado obtenido por la función de máxima entropía mejor estará funcionando el modelo. Existen otro tipo de funciones de pérdida que serán útiles dependiendo del problema.

$$D(S, L) = -\sum_{i}^{n} \log(S_i) * L_i \tag{A.3}$$

Estos vectores serán S con los resultados de la función softmax y L correspondiente a las etiquetas o *labels*. El vector L tendrá mismo tamaño que S y contiene todo ceros excepto por la clase a evaluar que tendrá valor 1, dicha clase a evaluar corresponderá con la de mayor probabilidad obtenida en la función softmax. Esta forma de representar el vector L recibe el nombre de **one-hot encoding**, visible en la Figura A.1. Agrupando todas diferentes capas explicadas en el modelo, se obtiene:

$$D(S(W \times X + B), L) \tag{A.4}$$

A.1.4. Precisión del modelo

Ahora, ya se conocen las probabilidades de que una imagen X pertenezca a las diferentes clases. Será necesario establecer una **función de pérdida** (*loss*, $\mathcal{L}$) que indique la precisión de nuestro modelo de manera numérica. En este caso se hará uso de la media de la función de máxima entropía en el conjunto de datos evaluado.

La función de pérdida deberá disminuir si la predicción $D(c, p)$, donde c es la clase real de la imagen y p la predicción obtenida por el modelo, es correcta $D(A, a)$ y aumentar si es incorrecta $D(A, b)$. La media de la función máxima entropía para el conjunto de imágenes de entrada para entrenar el clasificador recibe el nombre de *training loss*, si el conjunto de imágenes de entrada se usa para validar, recibe el nombre de *validation loss*.

$$\mathcal{L} = \frac{1}{n} \sum_{i}^{n} D(S(WX + B), L) \tag{A.5}$$

El objetivo será entonces tener el menor valor posible en la función máxima entropía para cada imagen, lo que marcará que las predicciones de las clases son correctas y la función de pérdida, por tanto, mínima. Puesto que la función de pérdida en esta arquitectura es dependiente de W y b, es posible minimizarla, teniendo ahora que resolver un problema de optimización.

A.1.5. Descenso de gradiente

Para obtener el mínimo W y b se hará uso del **algoritmo de descenso de gradiente** (GD). Dicho algoritmo actualiza el valor de los pesos W y el bias b de acuerdo a sus derivadas. Cada actualización de estos valores recibe el nombre de paso o *step*. Previamente a comenzar a iterar será necesario inicializar los pesos, este será un paso crucial a la hora de que la red entrene correctamente. Existen muchas técnicas como los autoencoders, compartir pesos con redes de menor tamaño o usar distribuciones como la uniforme para generarlos.

A la hora de calcular los gradientes, para el problema de minimización, estos son multiplicados por un término α llamado **coeficiente de aprendizaje** o *learning rate* (LR) que denota el grado de actualización de los valores de W y b en cada paso del algoritmo. Un valor excesivamente alto en el coeficiente de aprendizaje puede provocar que el paso dado por el algoritmo "salte" el mínimo global y por tanto no sea encontrado como se ve en el gráfico izquierdo de la Figura A.2, mientras que un valor muy bajo puede ralentizar el entrenamiento considerablemente incluso haciéndolo quedar atrapado en un mínimo local, tal y como se aprecia en la gráfica derecha de la Figura A.2.

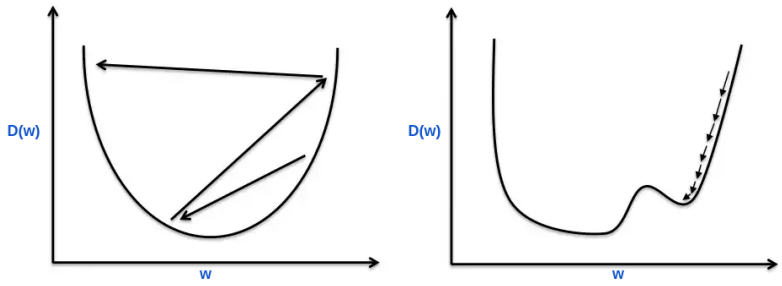


Figura A.2: Izquierda: problema con LR demasiado alto, salta el mínimo global y empeora. Derecha: problema con LR demasiado bajo, encuentra mínimo local y no avanza.

A la hora de entrenar la red serán necesarios un gran número de pasos del algoritmo de descenso de gradiente, generalmente, entrenar con todo el conjunto de datos provocará que el modelo actual tenga **overfitting**, es decir, que se haya adecuado demasiado a los datos que ha visto durante el entrenamiento haciendo que el uso del modelo con nuevas entradas no funcione tal y como lo hacía con los datos ya vistos, teniendo porcentajes de acierto mucho más bajos. Por ello, a la hora de entrenar una red neuronal el conjunto de datos se separa en dos o tres subconjuntos que servirán para

entrenar, validar o probar el modelo. El entrenamiento termina cuando los valores de W y b hacen que la función de pérdida o loss ($\mathcal{L}$) encuentre un mínimo global dentro del plano. Al observar que ($\mathcal{L}$) no mejora tras varias iteraciones, se asumirá que se encuentra en un mínimo global, por tanto, el entrenamiento habrá convergido.

$$W \leftarrow W - \alpha \times \Delta \mathcal{L}_W \tag{A.6}$$

$$b \leftarrow b - \alpha \times \Delta \mathcal{L}_b \tag{A.7}$$

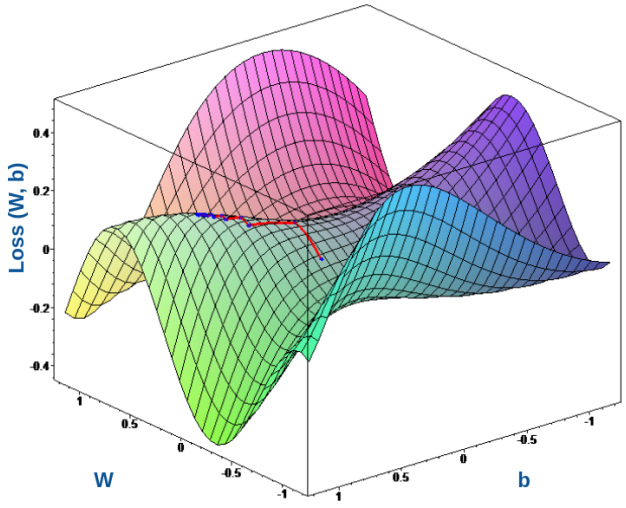


Figura A.3: Ejemplo del algoritmo de descenso de gradiente sobre las variables W y b. La zona coloreada son los valores que podría tomar la función de pérdida para dichos W y b, los puntos azules son los pasos del algoritmo de descenso de gradiente y las líneas rojas la dirección en la que se toman.

A.1.6. Descenso de gradiente estocástico

Llevar a cabo un paso del algoritmo de descenso de gradiente necesita tres veces más cálculos que obtener el valor de la función de pérdida, para cuyo cómputo se necesita el conjunto total de los datos, usando matrices que pueden tener tamaños considerables (con varios millones de parámetros), por lo que su coste computacional puede ser realmente elevado.

El descenso de gradiente estocástico hace uso de una pequeña parte de los datos de entrena-

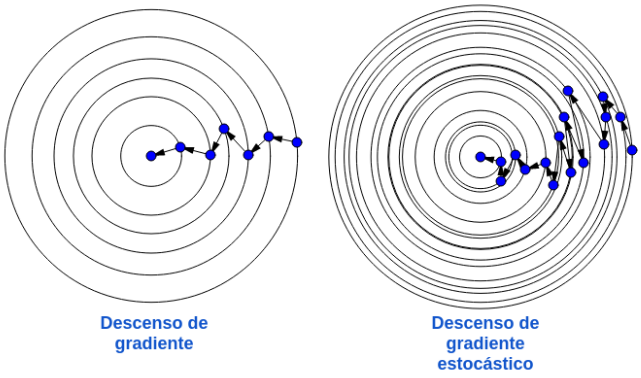


Figura A.4: Comparativa de pasos necesarios con el descenso de gradiente y el descenso de gradiente estocástico visto desde la planta.

miento (incluso una sola muestra) seleccionada aleatoriamente para hacer el cálculo de la función de pérdida mediante una estimación. Esto hará que los pasos a dar durante la ejecución del algoritmo sean relativamente rápidos, pero serán necesarios un mayor número de pasos hasta llegar al mínimo global.

A.1.7. Mejorando el entrenamiento

Al comienzo del entrenamiento de la red neuronal las variables W y b son capaces de aprender rápido (alto coeficiente de aprendizaje), pues aún no conocen el problema y lo más seguro se encuentren en un punto del plano alejado del mínimo global lo que les permite dar "pasos" mayores. Conforme el entrenamiento avanza los valores que W y b vayan adquiriendo en cada paso del algoritmo de descenso de gradiente harán que estén más cerca de alcanzar el mínimo global y converger.

Puede ocurrir, que, si se mantiene un coeficiente de aprendizaje alto, tal como se ha hecho al principio para acelerar el entrenamiento, los pasos dados en la actualización de las variables W y b por el algoritmo hagan que se salte directamente el mínimo global tal y como se ha visto en el gráfico izquierdo de la Figura A.2. Es por ello, que se usa una técnica conocida caída del coeficiente de aprendizaje (cuyo término en inglés es *learning rate decay*) que consiste en reducir el coeficiente de aprendizaje de manera que la actualización en cada paso de los valores de W y b sea menos pronunciada, provocando que cada vez se esté más cerca de converger sin llegar a saltar el mínimo global o pudiendo volver a alcanzarlo aunque se haya saltado, mejorando el entrenamiento tal como

se puede ver en el gráfico de la Figura A.5.

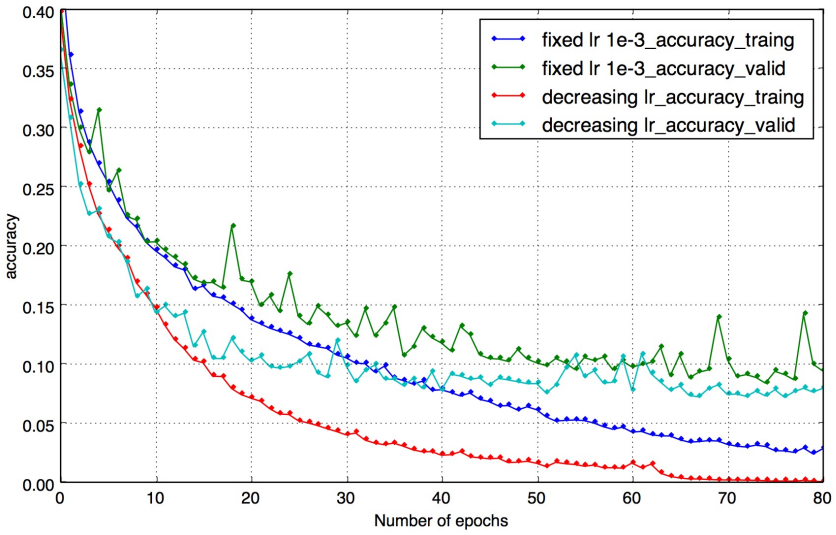


Figura A.5: Ejemplo de aprendizaje con caída en el coeficiente de aprendizaje vs coeficiente de aprendizaje fijo en entrenamiento y validación. Se observa como el modelo sin caída en el coeficiente de aprendizaje (azul y verde) se estanca en un 96% y 90% (1 − accuracy) de precisión para entrenamiento y validación respectivamente, convergiendo en el epoch 67, mientras que el modelo con caída del coeficiente de aprendizaje (rojo y cian) obtiene mejores resultados tanto en entrenamiento como en validación y converge más rápido, en el epoch 40[1].

Otra manera de mejorar el entrenamiento es hacer uso de **Momentum** (m), consiste en usar el conocimiento adquirido en cada paso del descenso de gradiente antes de dar el siguiente. Para ello se almacena la media de los gradientes calculados, de manera que al actualizar los valores de las variables W y b se tengan en cuenta los resultados anteriores. Esta técnica permite que las redes converjan antes y evita que el entrenamiento quede estancado en mínimos globales.

$$W(t+1) \leftarrow W(t+1) - \alpha \times \Delta\mathcal{L}_W(t+1) + m \times \Delta W(t) \tag{A.8}$$

$$b(t+1) \leftarrow b(t+1) - \alpha \times \Delta\mathcal{L}_b(t+1) + m \times \Delta b(t) \tag{A.9}$$

[1] Gráfica obtenida de https://adbrebs.wordpress.com

A.2. Redes neuronales profundas

Los modelos lineales, como el explicado en el Anexo A.1.1, son modelos estables que pueden ser útiles para tareas concretas pero poseen grandes limitaciones, dada su linealidad, que los hacen débiles a la hora de enfrentarse a problemas como los relacionados con la clasificación de imágenes. Las redes neuronales profundas buscan romper con dicha linealidad para abarcar un mayor número de tareas donde poder ser eficientes sin perder su estabilidad. Para conseguir esto, se introducirán funciones intermedias no lineales como las **Rectified Linear Units** (ReLus) [15].

$$f(x) = \max(0, x) \tag{A.10}$$

$$f'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{A.11}$$

Las ReLus hacen que los gradientes aprendan más rápido y por tanto la red converja antes. El modelo desarrollado tendrá ahora una capa intermedia que lo hará no-lineal. La arquitectura contará con dos funciones lineales con una ReLu intermedia. De la misma manera, siguiendo este proceso podremos añadir más capas no-lineales y lineales aumentando la profundidad del modelo.

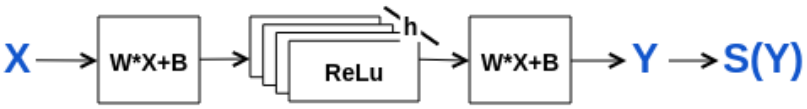


Figura A.6: Nueva arquitectura de la red con capas no-lineales.

Como se puede observar en la imagen, las ReLus tienen otro parámetro optimizable h, que será su profundidad. La profundidad afectará a su salida, haciendo que el número de filtros o canales varíe respecto a la imagen de entrada (normalmente, las imágenes de entrada tienen 1 canal si son en blanco y negro o tres si son en color).

Matemáticamente la complejidad del modelo no crece excesivamente gracias a la regla de la cadena que permite cambiar derivadas por productos y ejecutarlos de forma computacional con bajo coste dada una representación gráfica para el cálculo de las derivadas. Esta técnica de cálculo recibe

el nombre de *backpropagation*, y permite reusar los datos pre calculados para obtener los gradientes de manera eficiente.

A.2.1. Evitar el overfitting

Para evitar que al entrenar la red neuronal profunda esta incurra en overfitting existen un conjunto de técnicas. La primera a aplicar será la "terminación rápida" o *early termination* consiste en parar el entrenamiento de la red tan pronto como esta deje de mejorar, pues cada paso dado en el algoritmo de SGD llegado a este punto hará que el overfitting del modelo sea mayor.

Otra forma de mejorar la precisión será hacer uso de técnicas de regularización, esto consiste en añadir un término de penalización en el cálculo de la función de pérdida que actúe sobre W grandes.

$$\mathcal{L}' = \mathcal{L} + \beta \frac{1}{2} \|W\|_2^2 \tag{A.12}$$

Se considerará $\mathcal{L}'$ el nuevo valor del coeficiente de pérdida tras aplicar la regularización $L2$. Recibe el nombre de regularización $L2$ dado que se aplica la norma $L2$ con un coeficiente β al valor del coeficiente de pérdida. El modelo tendrá ahora otro hiper-parámetro a optimizar cuando se entrene, lo que beneficiará el resultado final a costa del tiempo necesario para encontrar el valor β óptimo.

Los datos transferidos entre capas reciben el nombre de activaciones, dichas activaciones son generalmente un vector con valores reales. La técnica del **dropout** [24] establece a 0.0 aleatoriamente una serie de estas activaciones. El porcentaje a eliminar será otro híper-parámetro a optimizar. El dropout permitirá tener varias representaciones con una misma entrada, por ejemplo: supongamos una entrada con 4 reales y un dropout de 0.5, si eliminas los índices 1 y 2 del vector obtenemos una salida de la siguiente forma $[0, 0, z, w]$ sin embargo si eliminamos los índices 3 y 4 obtendremos una salida diferente; $[x, y, 0, 0]$.

Se observa en la Figura A.7 que a pesar de obtener resultados similares en el conjunto de datos de entrenamiento (morado y azul), a la hora de validar el modelo con otro conjunto de datos completamente diferente, se tiene menor overfitting (amarillo con dropout y verde). Por lo que se concluye que la arquitectura representada por el morado y amarillo gracias al uso del dropout será más

[2] Gráfica obtenida de https://adbrebs.wordpress.com.

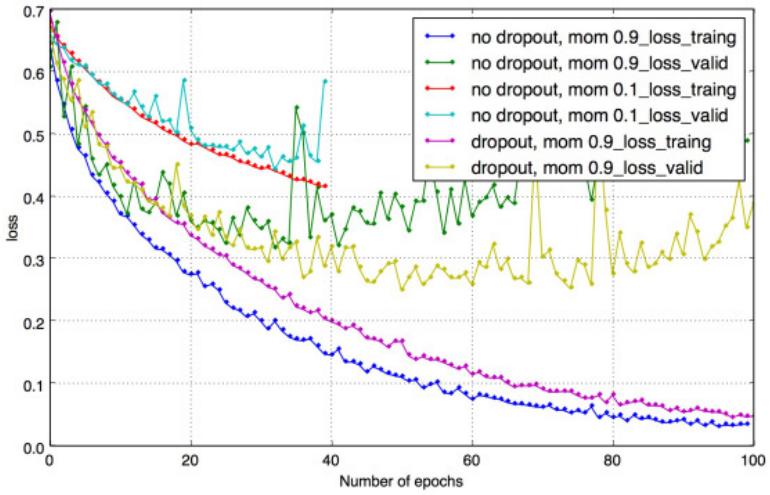


Figura A.7: Comparativa de un entrenamiento con y sin dropout, y con un momentum incorrecto. Como se puede observar las redes con buen momentum a pesar de obtener resultados similares en el conjunto de datos de entrenamiento (morado con dropout y azul), a la hora de validar el modelo (amarillo con dropout y verde), se observa cierto overfitting en la red que no ha hecho uso de dropout devolviendo peores resultados en el coeficiente de pérdida. Por lo que se concluye que la arquitectura representada por el morado y amarillo gracias al uso del dropout será más robusta al usarse con nuevos datos. La arquitectura inicializada con mal momentum es muy inestable haciendo imposible su entrenamiento[2].

robusta al usarse con nuevos datos. La arquitectura inicializada con mal momentum es muy inestable haciendo imposible su entrenamiento. La técnica del dropout, dado que usa aleatoriedad para frenar activaciones y establecerlas a 0 solo debe ser usado durante el entrenamiento. Mientras se evalúe el modelo esto no tiene que usarse, pues descartaría activaciones aleatorias haciendo que el modelo no fuera estocástico pudiendo generar predicciones diferentes para la misma entrada.

A.3. Redes neuronales convolucionales

Hasta ahora se ha visto la imagen de como una matriz de dos dimensiones, esto no tiene porqué ser así siempre, existen imágenes en color con 3 canales en cuyo caso serán matrices de tres dimensiones. Las redes neuronales convolucionales tienen una estructura similar a las neuronas de

la corteza visual primaria de un cerebro biológico y son más efectivas que un clasificador logístico o una red neuronal profunda para tareas relacionadas con las imágenes. Este tipo de redes aplican una convolución sobre la imagen de entrada considerando sus canales por separado. La convolución consiste en aplicar un filtro sobre parte de la imagen que, a modo de ventana deslizante, la recorrerá de acuerdo a unos parámetros dados como son el paso o *stride* y el *padding*. El stride hace referencia a los pixeles que avanza la ventana y el padding añadirá al principio y al final de la matriz filas y columnas de ceros para evitar perder dimensionalidad tras la convolución.

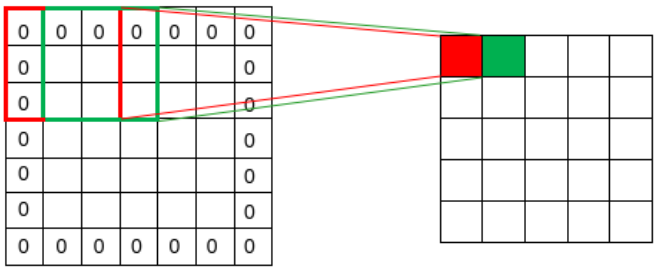


Figura A.8: Ejemplo de convolución de una imagen 5×5 con padding=1 y stride=1. Al tener 1 pixel de padding y 1 de stride, el resultado es una matriz del mismo tamaño a la inicial.

El filtro aplicado será la función lineal explicada al principio $W \times X' + b$ donde X' será la ventana, ahora el valor de W y b tendrá un tamaño acorde al tamaño de la ventana y no al de la imagen. Tal y como se puede ver en la figura A.8 se aplica una convolución a una imagen de 5 pixeles de alto y ancho con tan solo 1 canal de profundidad, la ventana tiene 1 pixel de stride y 1 de padding, por lo que tras la convolución la imagen resultante tendrá mismo tamaño que la inicial.

Al añadir un conjunto de capas con profundidad se obtiene una estructura piramidal como se observa en la Figura A.9. Asumiendo que no existe padding, trás cada convolución la matriz de entrada (imagen) se convierta en una matriz de menor altura y anchura, pero mayor profundidad, igual a la profundidad de los filtros aplicados.

La información de la imagen durante las convoluciones ha sido separada y transferida al siguiente nivel, de manera que las capas más cercanas a la imagen de entrada son capaces de identificar parámetros de bajo nivel como son colores y bordes, mientras que las capas más alejadas son capaces de reconocer formas u objetos de más alto nivel y hasta entidades completas. Por lo tanto, lo que

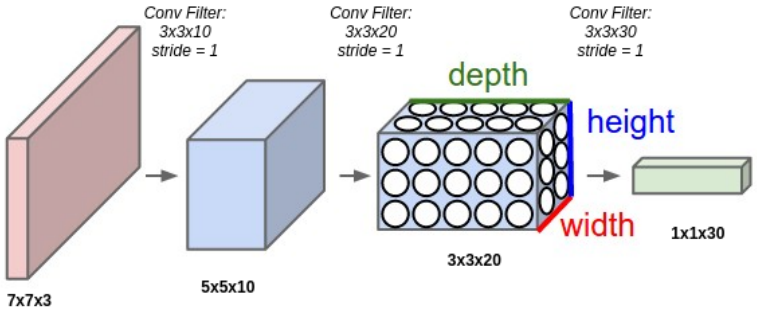


Figura A.9: Arquitectura de una red neuronal convolucional con 3 capas de convolución.

se pretende es clasificar la matriz última de convolución. La arquitectura de la red convolucional se completa añadiendo un clasificador lineal simple, como el visto al comienzo de este Anexo A.1.1, que pueda diferenciar de acuerdo a los parámetros de las últimas capas a qué clase corresponde dicha entrada.

Para poder realizar el entrenamiento de este tipo de redes se requiere de potentes GPUs. Las imágenes de entrada pueden tener tamaños medianos y los filtros pueden ser ligeramente profundos haciendo que sean necesarios cálculos para millones de parámetros solo en una capa (algunas redes alcanzan más de 20 capas y 150 millones de parámetros).

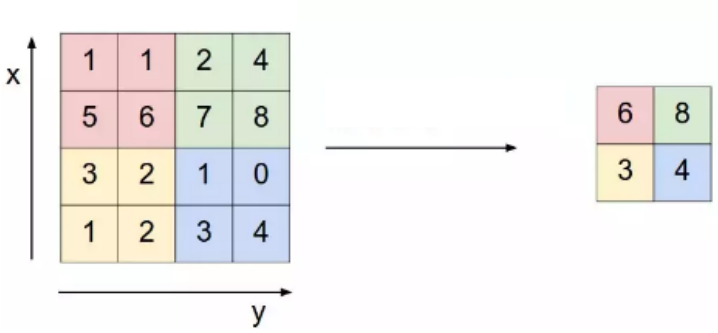


Figura A.10: Max pooling sobre una imagen de 8×8 píxeles, con un filtro 2×2 y stride 2.

Existen técnicas que mejoran los tiempos de computación y la memoria usada, como, por ejemplo, el *pooling* que agrupar un conjunto de píxeles aplicándoles una función como puede ser el máximo o la media, reduciendo la dimensionalidad de la imagen y controlando el overfitting. El ejemplo más común de pooling es el usado sobre un filtro de 2×2 de la imagen con stride de 2 píxeles tal y como

puede apreciarse en la Figura A.10

B INSTALACIÓN Y DESARROLLO

Al comienzo del proyecto se tuvo que estudiar qué entorno de trabajo usar, así como lenguaje y con qué red neuronal comenzar a trabajar. Se exploró el ecosistema actual del Deep Learning, así como las posibles herramientas posibles de desarrollo, lenguajes y respaldo dado por otros laboratorios o grupos de investigación. La Sección B.1 comenta las decisiones tomadas al comienzo del proyecto, el framework usado así como las librerías necesarias, seguidamente, la Sección B.2 explica el entorno de desarrollo empleado.

B.1. Framework y librerías empleadas

Finalmente, tras una exploración del ecosistema actual del Deep Learning se optó por hacer uso de **Torch**[1] [3] dada su sencillez y eficacia (además del respaldo dado por grupos como Twitter, Google DeepMind, Facebook AI o Yandex). Junto con Torch son necesarias un conjunto de librerías que hacen posible realizar operaciones con tarjetas gráficas, y por tanto acelerar el proceso de entrenamiento o evaluación de redes neuronales.

loadcaffe Permite cargar modelos pre entrenados de Caffe (otro framework de Deep Learning

[1] Proyecto Torch: http://torch.ch/

en Torch ejecutándolos como si fueran uno propio, con toda su funcionalidad.)

cutorch y cunn Conjuntamente con CUDA permiten realizar operaciones con la GPU ahorrando mucho tiempo en cálculos con grandes matrices.

cudnn Conjunto de operaciones asociadas con CUDA, implementadas de manera muy eficiente (convoluciones o poolings) que aceleran los cálculos, haciéndolas realmente útiles en Deep Learning.

hdf5 Librería que permite el uso del formato de ficheros HDF5 en torch, permitiendo gran flexibilidad y capacidad de comunicación entre aplicaciones.

nn Aporta las funciones de las redes neuronales. Hace uso de funciones modulares fácilmente escalables permitiendo crear estructuras de redes neuronales complejas.

optim Paquete que contiene diferentes algoritmos y rutinas de optimización, además de ofrecer una utilidad para poder registrar y mostrar gráficamente los resultados.

matio Librería que permite cargar ficheros *.mat* con estructuras de datos complejas, así como, escribirlos almacenando objetos creados con Torch.

Conjuntamente con Torch se hace uso de **Python**[2] para las tareas relativas a las máquinas de soporte de vectores (SVM). Python no implementa en su núcleo las funciones necesarias para la manipulación de SVM o lectura de ficheros HDF5, por ello es necesario instalar una serie de paquetes que hacen posible su uso:

h5py Interfaz para hacer uso del formato de datos HDF5 en Python, funciona de manera simple haciendo que su uso sea inmediato.

sklearn Incluye un conjunto de funciones y algoritmos del aprendizaje automático. Se pueden encontrar herramientas para la clasificación, reducción de dimensionalidad, regresión, selección de modelos, clustering y pre procesamiento entre otros.

[2]Web de Python https://www.python.org/

numpy Paquete base para realizar cálculos y operaciones con matrices y estructuras multidimensionales. Además, incluye funciones del álgebra lineal y herramientas que integran C/C++ y código Fortran.

B.2. Entorno de trabajo

Tras conocer el framework a usar y haber comenzado con el estudio teórico del actual estado del arte en la tarea de clasificación de imágenes, así como la creación de arte con redes neuronales. Se decide comenzar a buscar un editor de textos que sea funcional con Torch. Finalmente se hace uso de **Atom**[3] que permite hacer uso de torch gracias al kernel iTorch[4] basado en iPython que permite compilar y ejecutar código de manera modular. Atom es un editor de texto configurable y flexible, que funciona gracias a la instalación de paquetes, dada su posibilidad de interactuar con kernels es posible usarlo para desarrollar diferentes lenguajes, pudiendo hacer uso de los algoritmos desarrollados en Torch y Python desde el mismo editor, además de escribir esta memoria con LaTeX sin necesidad de cambiar de entorno.

Para poder hacer uso de Atom con todos los entornos es necesaria la instalación de la librerías **iTorch**, **iPython** y la distribución **TeX Live**. Además, son necesarios los siguientes paquetes en Atom:

hydrogen Permite la interacción con un gran número de kernels donde se incluyen iTorch y iPython. Gracias a este paquete es posible compilar y ejecutar modularmente el código, siempre que se disponga de su kernel en cuestión.

language-(lenguaje) (lenguaje) será el lenguaje de programación del cual queremos instalar su sintaxis para que sea reconocida en Atom.

linter-(lenguaje) Similar al paquete *languaje*, su función es resaltar posible texto a autocompletar del (lenguaje) instalado.

pdf-view Permite la visualización de archivos pdf. Paquete útil para hacer uso mientras se escribe en LaTeX y poder visualizar los cambios realizados.

[3]Web del proyecto Atom https://atom.io/

[4]Web de iTorch https://github.com/facebook/iTorch

Conocidas todas herramientas a usar, se decide tomar como punto de partida el código disponible del algoritmo neural-style [4][5] del que finalmente solo se hace uso del modelo pre entrenado de la red VGG Net 19 [23]. Durante el desarrollo de los algoritmos se usa **GitHub** para poder llevar un control de las versiones, tanto del código como de la memoria. Además, para poder marcar las tareas realizadas, a hacer y en proceso se emplea **Trello**, una web con una interfaz de usuario amigable que implementa el método Kanban, donde se puede observar que se ha realizado cada día, compartir archivos y comentar nuevas investigaciones que pudieran ser interesantes. A la hora de trabajar remotamente (dada la necesidad de la potencia de la máquina situada en el laboratorio y la imposibilidad de estar en este todo el tiempo) se hizo de uso de **ssh** para enviar comandos puntuales y del software **teamviewer** para poder trabajar completamente en remoto.

[5]Neural-style https://github.com/jcjohnson/neural-style

CONTENIDO DEL REPOSITORIO

El repositorio contenido en GitHub con el trabajo realizado se compone de 3 carpetas principales:

data Compuesta por tres subcarpetas:

- La primera, con nombre data_utils, contiene los ficheros necesarios para la correcta ejecución del algoritmo como la relación entre el vector de probabilidades devuelto por la red y la clase que representa.
- La carpeta results contiene los resultados devueltos por los modelos probados.
- Por último, se encuentra la carpeta paths con datos relativos a las rutas de las imágenes.

Además de las subcarpetas, hay un fichero correspondiente a la aplicación usada para obtener imágenes haciendo web-scrapping.

models Dicha carpeta contiene los modelos probados pre entrenados así como la red VGG Net 19 [23], dado su peso (más de 1 GB e incluso 2.5 GB dependiendo del modelo) esta carpeta se encuentra vacía

src Contiene el código usado para la clasificación de imágenes. En su raíz se encuentra el archivo *main* que el archivo principal para la ejecución del programa y otro

fichero *feature_extract* que es una utilidad para extraer activaciones de una capa de una carpeta con un conjunto de imágenes. Se pueden encontrar también cuatro subcarpetas.

- La primera con nombre SVM tiene el código necesario para entrenar y evaluar las SVMs propuestas.
- La segunda subcarpeta, con nombre autoencoder, tiene el código creado para poder entrenar y evaluar un auto codificador (no se llegó a usar).
- La tercera carpeta cuyo nombre es fine-tune, tiene el código usado para reentrenar y evaluar nuevos modelos.
- Por último, la carpeta utils, contiene un conjunto de ficheros con funciones que simplifican tareas llevadas a cabo frecuentemente

En la raíz se pueden encontrar también un archivo INSTALLATION relativo a instrucciones para la instalación, y diversos ficheros usados por GitHub para llevar a cabo su control de versiones.

D RESULTADOS ADICIONALES

El siguiente capítulo contiene gráficas que muestran resultados adicionales no incluidos en el contenido principal de la memoria.

La Figura D.1 muestra los resultados sobre los conjuntos de datos Noisy 23 y Curated de la red durante su optimización en el epoch 100.

La Figura D.2 muestra los resultados sobre los conjuntos de datos Noisy 23 y Curated de la red durante su optimización en el epoch 200.

La Figura D.3 muestra los resultados sobre los conjuntos de datos Noisy 23 y Curated de la red durante su optimización en el epoch 100, sustituyendo la capa softmax y la última fully connected (1000 activaciones) por una SVM.

La Figura D.3 muestra los resultados sobre los conjuntos de datos Noisy 23 y Curated de la red durante su optimización en el epoch 200, sustituyendo la capa softmax y la última fully connected (1000 activaciones) por una SVM.

La Figura D.5 muestra la suma de las diferencias al cuadrado de los pesos en la red VGG Net 19 [23] y esta misma optimizada en el epoch 200. Se observa que la red tiene hasta 35 capas, mientras que en la Sección 4.1 se nombraba que esta capa contenía 19 capas con pesos. Esto se debe a que se encuentran incluidas las capas sin pesos en la leyenda sobre el eje x.

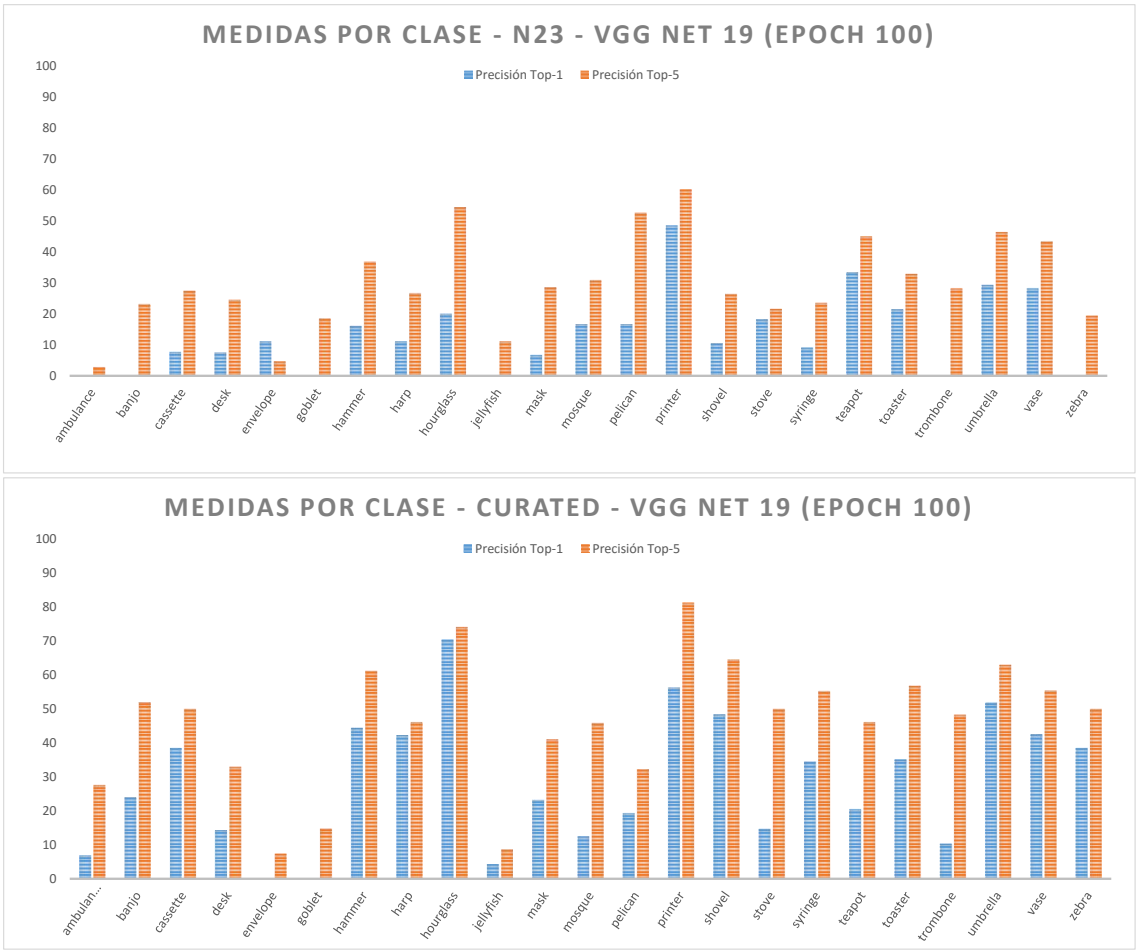


Figura D.1: Las Figuras (a) y (b) muestran los resultados durante el entrenamiento para los conjuntos de datos noisy y curated respectivamente en el Epoch 100, usando la capa softmax para la clasificación de las imágenes.

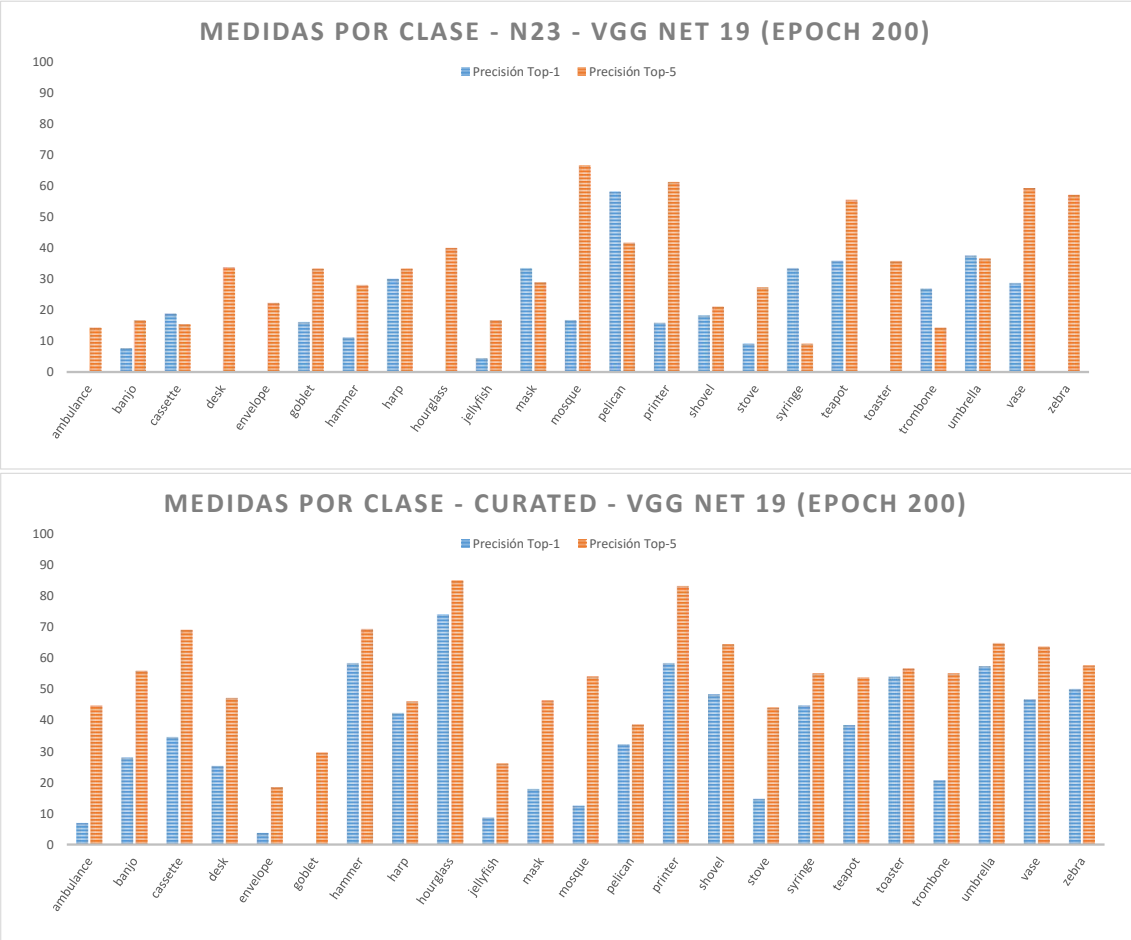


Figura D.2: Las Figuras (a) y (b) muestran los resultados durante el entrenamiento para los conjuntos de datos noisy y curated respectivamente en el Epoch 200, usando la capa softmax para la clasificación de las imágenes.

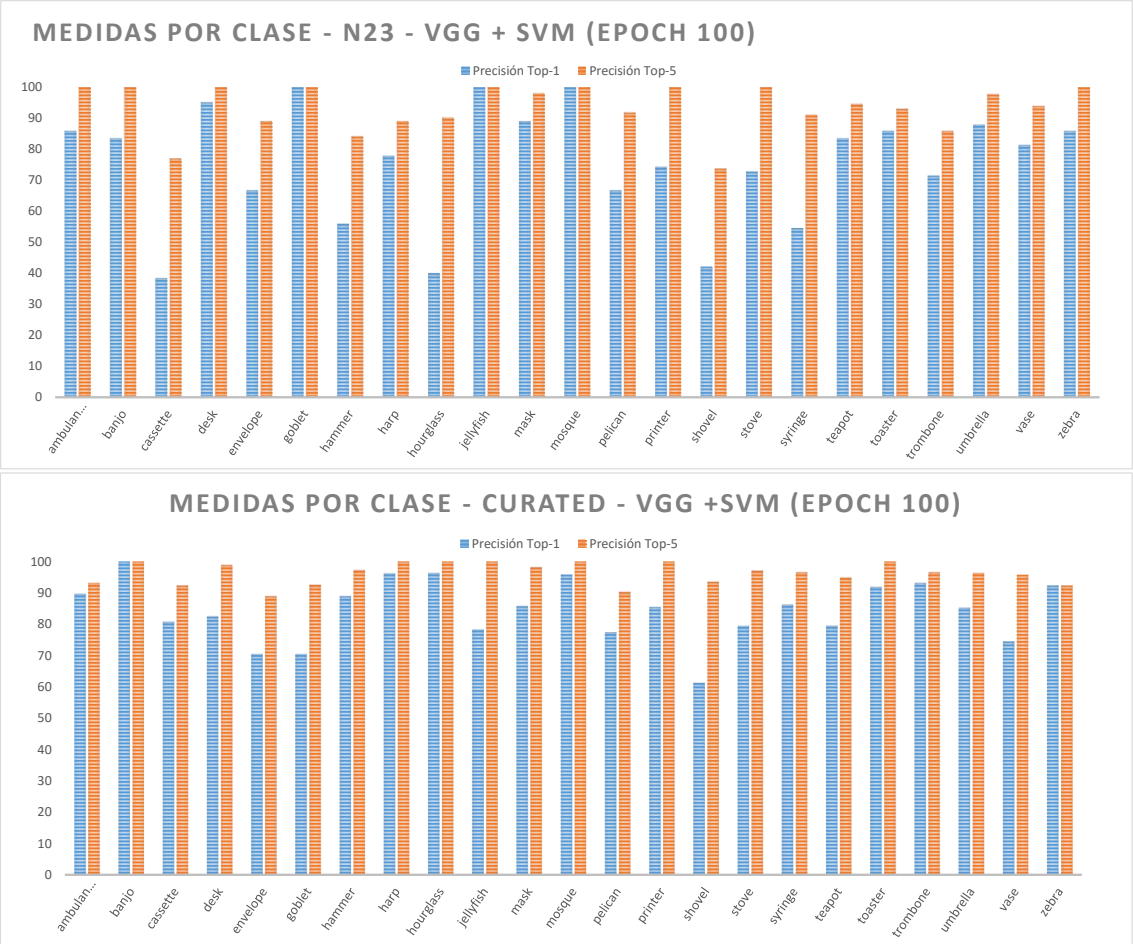


Figura D.3: Las Figuras (a) y (b) muestran los resultados durante el entrenamiento, concretamente en el Epoch 100, para los conjuntos de datos noisy y curated respectivamente haciendo uso de una SVM como su clasificador.

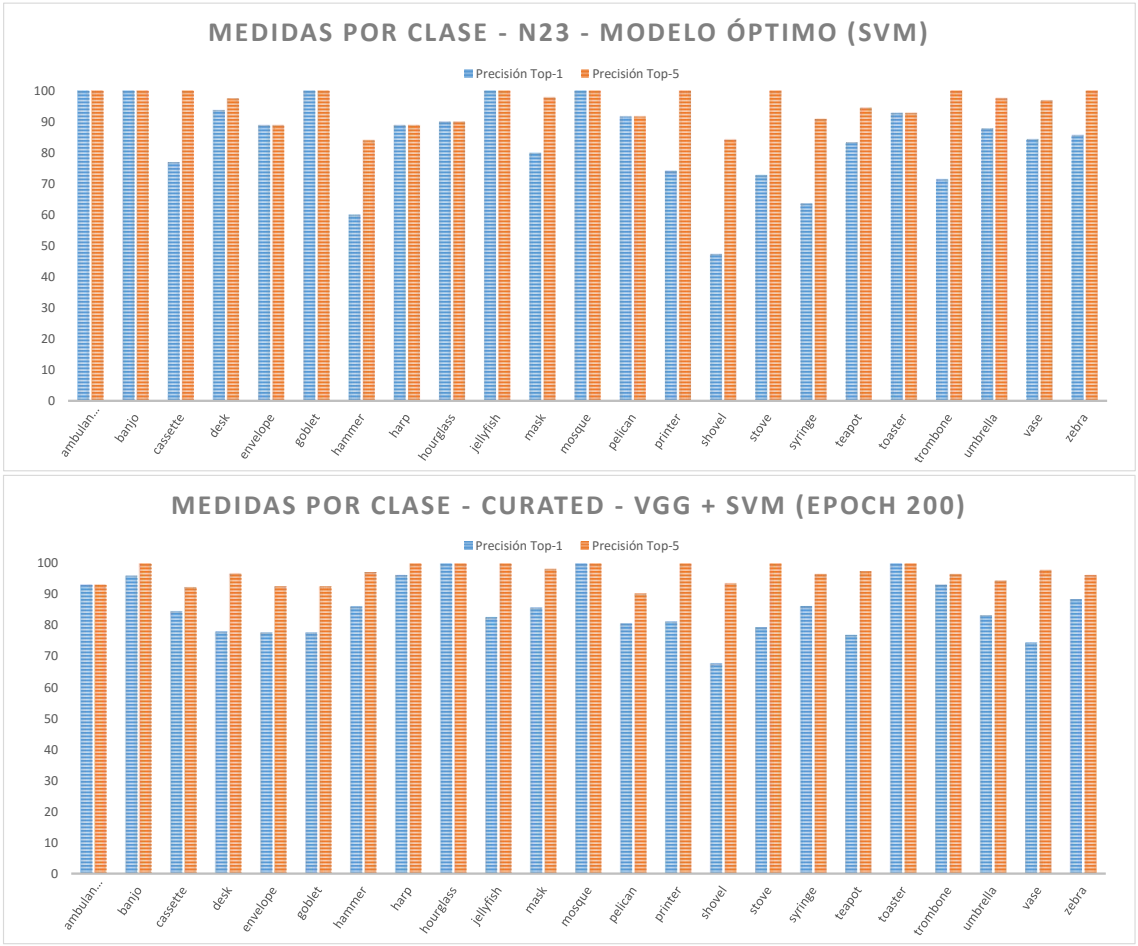


Figura D.4: Las Figuras (a) y (b) muestran los resultados durante el entrenamiento, concretamente en el Epoch 200, para los conjuntos de datos noisy y curated respectivamente haciendo uso de una SVM como su clasificador.

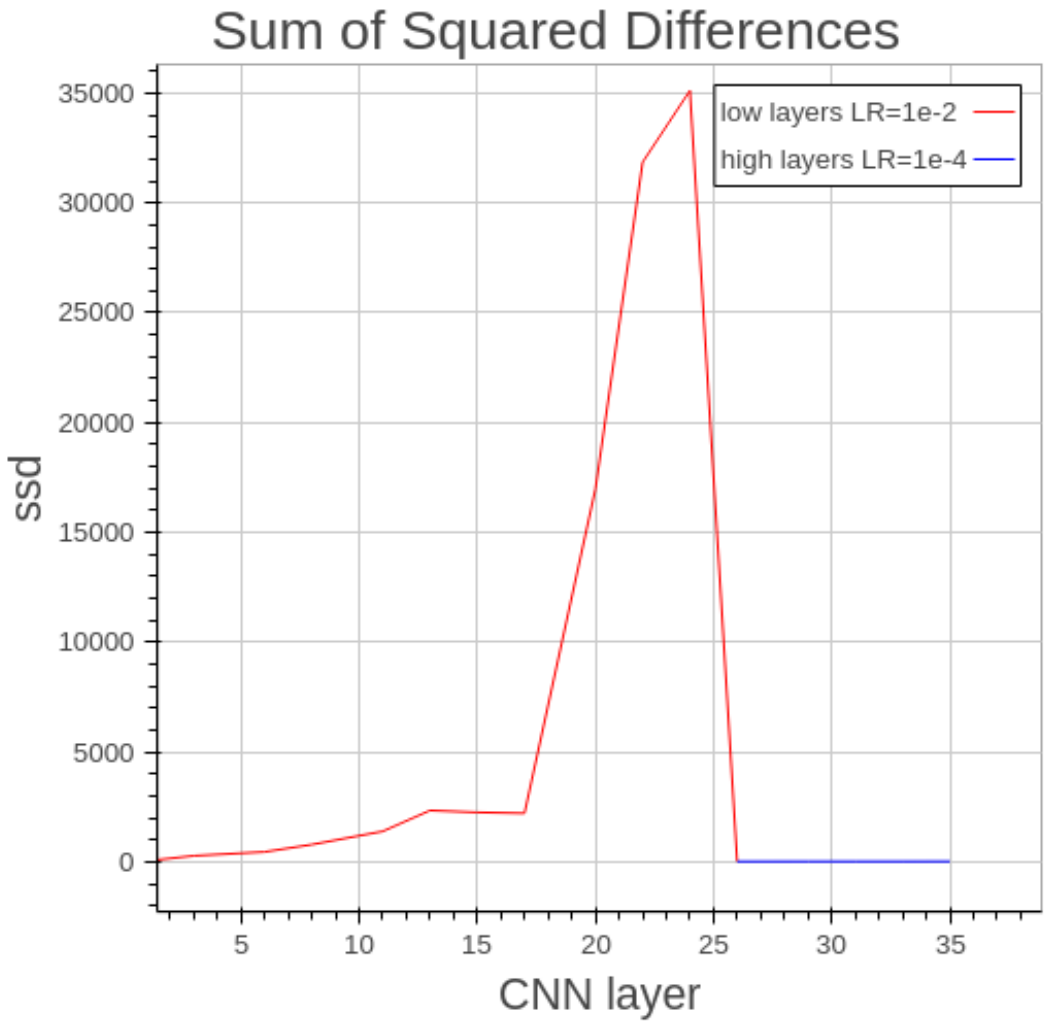


Figura D.5: Suma de las diferencias al cuadrado (SSD) de los pesos en la red VGG Net 19 y la misma red tras 200 epochs de optimización.