E.T.S. de Ingeniería Industrial,

Informática y de Telecomunicación

Estudio del impacto del tamaño de ventanas en Deep Learning

Grado en Ingeniería Informática

Trabajo Fin de Grado

Alumna: Izaskun Rodríguez Villar

Director: Fco. Javier Fernández Fernández

Co-director: Javier Fumanal Idocin

Pamplona, 21 de Octubre de 2020

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Índice general

1.Resumen

Las redes neuronales convolucionales (Convolutional Neuronal Networks or CNN) son hoy en día la técnica más popular para realizar el tratamiento de imagen. Estas redes están formadas principalmente por dos capas: filtros convolucionales y reducción de imagen (pooling). En este trabajo vamos a evaluar el impacto en el rendimiento de una red de distintos parámetros inherentes a la construcción de la misma: el tamaño de la ventana de convolución, el tamaño de la ventana de pooling, el uso de distintas funciones para la reducción de imagen (la más habituales son el máximo y la media), etc. Con el objetivo de calcular que configuraciones son mejores y si existe una diferencia significativa entre ellas. Una vez se tengan las diferentes configuraciones, se calcularán los resultados de acierto para cada una de ellas y se comprobará si hay diferencia estadística entre ellas.

2.Palabras clave

Deep Learning, Redes Neuronales Convolucionales, Filtros convolucionales, Capa de pooling.

3.Introducción

El procesamiento de datos es una herramienta clave para obtener información útil con el fin de proponer conclusiones y apoyar en la toma de decisiones. Desde la aparición del computador es posible trabajar con cantidades masivas de datos en el ámbito de la extracción y el análisis de ellos. Las técnicas de Inteligencia Artificial han permitido utilizar estos datos para poder tratar de forma masiva y automática, con una potencia de cálculo suficiente, la información situada en ficheros de datos almacenados.

El término Inteligencia Artificial se definió por primera vez en los años cincuenta por John McCarthy [13]. Una buena definición de ella puede expresarse como la simulación de la inteligencia humana por un computador, con el fin de hacer que la máquina sea capaz de identificar y utilizar el "conocimiento" en una etapa determinada de la solución de un problema planteado. Cabe diferenciar entre Inteligencia Artificial débil y fuerte. La IA débil, es aquella inteligencia artificial y razonable que se centra únicamente en tareas limitadas. Por ejemplo, se podría indicar que se trata de la capacidad para adquirir y aplicar conocimiento, o la facultad de pensar y razonar. Ya en 1950 Turing [14], en un artículo para la revista Mind (Computing Machinery and Intelligence) propuso una definición operacional basada en un test para identificar la inteligencia de una máquina. Esta superaba el test de inteligencia, si un ser humano, después de plantearle algunas cuestiones, no podía discriminar si las respuestas obtenidas procedían o no de una persona. En cambio, la IA fuerte es aquella capaz de igualar o exceder la inteligencia de los humanos.

En 1958 Frank Rosenblatt diseña el perceptrón, también conocido como la primera red neuronal artificial [1]. Este modelo es conocido como perceptrón simple el cual es un modelo monocapa de neurona basado en el modelo de McCulloch y Pitts y en una regla de aprendizaje basada en la corrección del error. Se caracterizó por su capacidad de aprender a reconocer patrones. El perceptrón está constituido por un conjunto de sensores de entrada que reciben los patrones de acceso a reconocer o clasificar y una neurona de salida que se ocupa de catalogar los patrones de entrada en dos clases, según si la salida obtenida es 1 (activada) o 0 (desactivada). Sin embargo, dicho modelo tenía muchas limitaciones, como por ejemplo, no era capaz de aprender la función lógica XOR [2].

El perceptrón multicapa apareció en el año 1969. A diferencia del unicapa, se demostró que era un aproximador universal a causa de la regla de aprendizaje de retropropagación del error y surgió como consecuencia de las limitaciones de la arquitectura en lo referente al problema de la separabilidad no lineal [3]. Dando lugar a la generación de un nuevo tipo de red, cuya modificación principal respecto a la estructura del perceptrón se basa en el uso de varias capas de neuronas artificiales, en lugar de usar una sola capa. Tanto el perceptrón simple como el multicapa son clases de redes neuronales o también conocidas como redes con propagación hacia delante. Al combinar ambos perceptrones, se pudo contribuir en algunos estudios posteriores realizados por Rumelhart, Hinton y Willians en 1986 [4].

En la segunda mitad de la década de los 70 el campo de la IA sufrió su primer "invierno" debido a que los descubrimientos hechos hasta entonces no habían cumplido las expectativas prometidas, y muchos fondos para la investigación fueron suprimidos [5].

En 1986 Rina Dechter [6] propone el término "Deep Learning" (Aprendizaje Profundo). Se trata de un algoritmo automático que imita la forma de "ver" del ser humano inspirada en nuestro cerebro y la conexión entre neuronas. Esta es la técnica que más se acerca a la forma en la que aprendemos los humanos. El deep learning se conoce como "redes neuronales profundas" o "deep neural networks". Es denominado como "deep" en referencia a las capas que tienen estas redes neuronales [7]. Han llamado mucho la atención por su potencial utilidad en distintos tipos de aplicaciones en el "mundo real" [8] (puede aplicarse a casi cualquier técnica [9], haciéndolo muy útil en su aplicación a imagen y voz), principalmente debido a que obtiene tasas de éxito elevadas con entrenamiento "supervisado", este requiere de una potencia de cálculo significativa, por eso las GPU de alto rendimiento que tienen arquitectura paralela resultan eficientes para ello. Un ejemplo donde se puede usar es para abordar problemas en Big Data Analytics [10], así mismo también se puede usar en problemas de imagen [11] y reconocimiento de voz [12].

Aunque existen numerosas arquitecturas de red y los algoritmos de la IA son aplicados con mucho éxito en el procesamiento de imagen, teniendo en cuenta los requisitos de este campo, para obtener resultados óptimos en cualquier problema planteado, no parece hallarse un algoritmo para decidir qué arquitectura es la mejor para un determinado problema. Este es precisamente el problema que vamos a considerar.

Más específicamente, en este trabajo estudiaremos el efecto de diversos parámetros sobre el rendimiento de una CNN, siendo nuestro objetivo obtener la mejor combinación de parámetros para un problema de clasificación a partir de imágenes.

4.Objetivos

El objetivo del proyecto consiste en evaluar el impacto en el rendimiento de una red neuronal convolucional en el tratamiento de imagen de distintos parámetros inherentes a la construcción de la misma. Para ello se partirá del conjunto de datos CIFAR-10. Con el fin de conseguir dicha meta tendremos que:

- Calcular cuáles son las mejores configuraciones que se comportan con este conjunto de datos.
- Encontrar si existen diferencias significativas entre las configuraciones.
- Conseguir las mejores configuraciones.

5.Estado del arte

En esta sección vamos a hablar de las redes neuronales tradicionales (perceptrón, perceptrón multicapa) usadas típicamente en problemas de clasificación, y de las redes convolucionales (CNN). Generalmente, el término Deep Learning en imagen se aplica a CNN cuyas capas imitan al córtex visual del ojo humano para identificar distintas características en las entradas que en definitiva hacen que pueda identificar objetos y "ver".

5.1.Perceptrón

El perceptrón simple fue introducido por Frank Rosenblatt en 1958 [16]. Es reconocido por su capacidad de aprender a detectar patrones [17]. Su arquitectura es muy sencilla, se compone de una red monocapa con una única neurona de entrada conectada a la neurona de salida. Sin embargo, este modelo tiene el problema de que no es un estimador universal.

El perceptrón multicapa es una evolución del perceptrón simple que incorpora capas de neuronas ocultas, con esto consigue representar funciones no lineales. Está compuesto por una capa de entrada, una capa de salida y n capas ocultas intermedias [18] como se puede observar en la Figura 1.

En todas las redes neuronales podemos diferenciar dos fases:

1. **Propagación:** En la que se calcula el resultado de salida de la red desde los valores de entrada hacia delante.
2. **Aprendizaje:** En la que los errores obtenidos a la salida del perceptrón se van propagando hacia atrás (**backpropagation**), esta aproximación se realiza mediante la función gradiente del error.

5.2.Redes Neuronales

Una red neuronal o red neuronal artificial [15], es un modelo inspirado en el funcionamiento del cerebro humano.

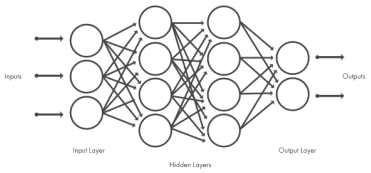


Figura 1: Red Neuronal Artificial

Para explicar su funcionamiento nos basaremos en la Figura 1. Estas neuronas están agrupadas en capas donde cada una de ellas se conecta con todas las de la capa siguiente. Las neuronas que se encuentran en la siguiente capa reciben una media ponderada de las salidas de la capa anterior (1). A dicha salida se le aplica una función no lineal, denominada función de activación, para obtener el nuevo resultado.

$$y_i = \sum_{i=1}^{n} w_{ij} x_j - \theta i \tag{1}$$

Donde x_j es un conjunto de entradas, w_{ij} unos pesos sinápticos, con j=1,...,n y θi el umbral.

El resultado de esta ecuación es una entrada para las neuronas de la siguiente capa.

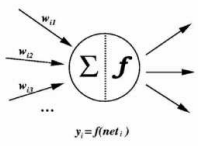


Figura 2: Ejemplo del modelo neuronal

Dependiendo de la red neuronal con la que trabajemos, las funciones de activación serán de un tipo u otro. De todas ellas, la función más utilizada es la función sigmoide (Figura 3).

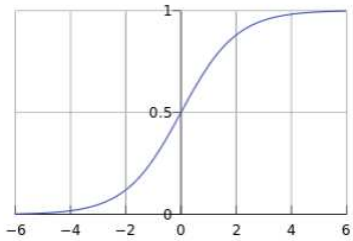


Figura 3: Función Sigmoide

Las neuronas se agrupan en tres tipos de capas:

- Capas de entrada: En esta capa no hay procesamiento, las neuronas introducen los valores de entrada en la red, por lo que se tendrán tantas neuronas como componentes tengan los datos de entrada.
- Capas ocultas: Esta capa no tiene conexión directa con el entorno, puede estar seguida de otra capa oculta o bien de la capa de salida. El número de capas ocultas puede ser variable en cada caso.
- Capas de salida: En esta capa cada neurona que la compone tiene asociado un peso y un parámetro. El valor de cada una de ellas corresponde a el valor de salida da la red neuronal.

En esta arquitectura, el proceso de aprendizaje se lleva a cabo basándose en una serie de ejemplos de entrada de los cuáles sabemos su salida correspondiente. Este entrenamiento le permite posteriormente relacionar entradas y salidas para datos desconocidos.

5.3.Redes Neuronales Convolucionales

Las redes neuronales profundas más populares son las redes neuronales convolucionales (CNN) [19]. La peculiaridad de estas redes es que cada neurona de una capa sólo recibe conexiones de algunas de las neuronas de la capa anterior.

Se denomina convolución clásica a un operador matemático que transforma dos funciones, f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g.

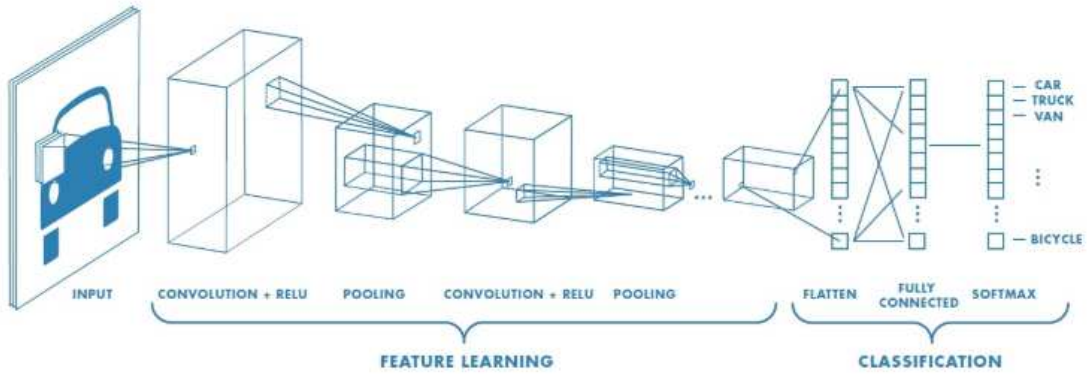


Figura 4: Esquema genérico de una red neuronal convolucional

Cabe destacar una diferencia entre las CNN y las redes neuronales tradicionales:

- Las CNN tienen un número de capas elevado, así que es necesario tener en cuenta las capacidades de GPU del dispositivo que se va a utilizar a la hora de llevar a cabo el proceso de entrenamiento y de aplicarlas.

En la Figura 4 se muestra una imagen de la estructura básica de un CNN [20].

Como se puede ver en dicha figura, la red convolucional se compone de las siguientes capas:

- Capa de convolución: Es la capa oculta donde se define para cada neurona los parámetros de ajustes (o bias), los filtros y la función asociada. Esta incluye la función ReLU también llamada función de activación. Es decir, ella procesará la salida de neuronas que están conectadas en "regiones locales"de entrada, calculando el producto escalar entre sus pesos y una pequeña región a la que están conectadas en el volumen de entrada. En la Figura 5 observamos la estructura de dicha capa.

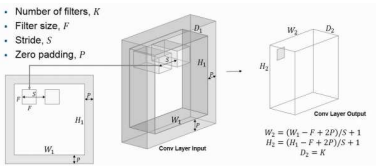


Figura 5: Capa de convolución [21]

- Capa de pooling: Esta capa se utiliza para reducir la dimensión de la imagen. Las capas de pooling más utilizadas son max-pooling y average-pooling. En la Figura 6 observamos ejemplos de pooling.

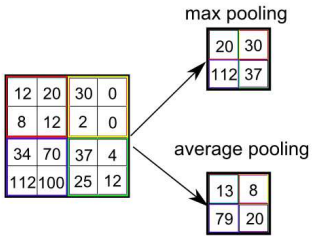


Figura 6: Ejemplos de pooling en una CNN [22]

- Capa fully connected (FCL): Es la red neuronal completa que se conecta a la última capa de la red. Esta necesita una adaptación intermedia para convertir los datos a vector (capa Flatten) como podemos ver en la Figura 4. Por último, tenemos la capa Softmax que se encarga de asociar una etiqueta a la imagen de entrada atribuyendo una probabilidad de pertenencia a cada clase.

Por otra parte, hay que tener en cuenta las distintas funciones de activación a elegir. Entre las más comunes se encuentran:

- **Sigmoide:** La función sigmoide o σ (representada gráficamente en la Figura 3), es usada ante todo para las capas de salida. Se define de la siguiente manera:

$$f(z) = \frac{1}{1 - \epsilon^{-z}} \tag{2}$$

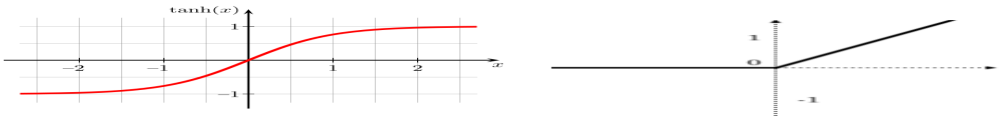
- **Tanh:** La función tangente hiperbólica o tanh, es una función de activación recomendada para su uso en las capas ocultas, a veces es útil para las capas de salida. En la Figura 7 (a) se ve su representación gráfica. Se define de la siguiente manera:

$$f(z) = \frac{\epsilon^{z} - \epsilon^{-z}}{\epsilon^{z} + \epsilon^{-z}} \tag{3}$$

- **ReLU:** La Rectified Linear Unit, o ReLU, es recomendada para usarla en las capas ocultas de la red. En la Figura 7 (b) se ve su representación

gráfica. Se define como:

$$f(z) = \text{máx}(0, z) \tag{4}$$



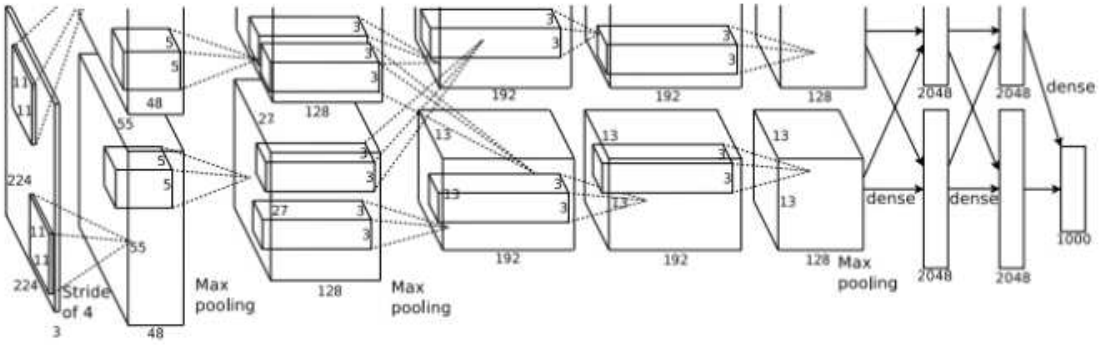
(a) Función Tanh (b) Función ReLU

Figura 7: Representación de funciones

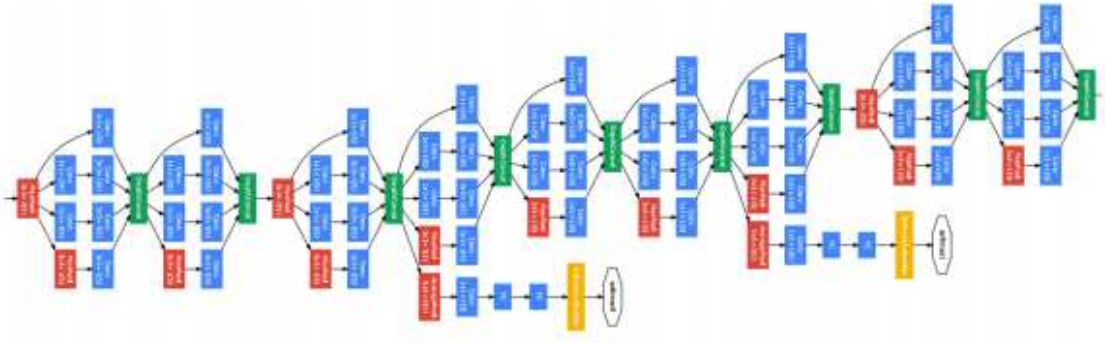
- **Softmax:** Esta función se utiliza en casos de varias neuronas de salida para obtener una clasificación multiclase. Su la salida es una distribución de probabilidad donde la salida de todas las neuronas suma 1.

Finalmente, cabe mencionar algunas de las arquitecturas de CNN más comunes [23]:

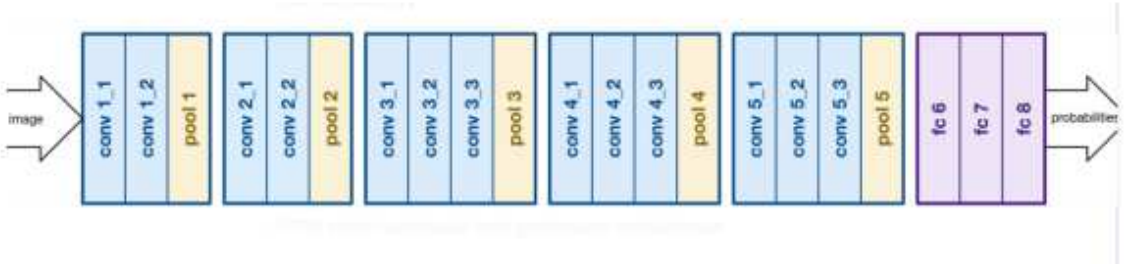
- **AlexNet:** Fue propuesta por Alex Krizhevsky en 2012. Es la primera arquitectura de redes neuronales que se estandariza tras la participación de la misma en el ILSVRC (ImageNet Large Scale Visual Recognition Challenge). En esta red se utilizó la función ReLU, que mostró un mejor rendimiento de entrenamiento que la tangente hiperbólica y la sigmoide [24]. Se puede observar esta arquitectura en la Figura 8 (a).
- **GoogLeNet:** Fue creada por Christian Szegedy en 2014, consiguió una reducción del número de parámetros de la red. Esta mejora se basó en la modificación de la arquitectura de la red, en vez de terminar la red con una fully conected layers se utilizó una capa de pooling utilizando el método de average [25]. Esta nueva red tiene 4 millones de parámetros, frente a la AlexNet que tiene 60 millones de parámetros, esto supone una notable mejora y el proceso de entrenamiento será mucho menos costoso. Dicha arquitectura la observamos en la Figura 8 (b).
- **VGGNet:** Con este modelo propuesto por Simonyan y Zisserman en 2014, se demostró que la profundidad de la red es una parte fundamental para obtener mejores resultados. Esto supone un mayor número de parámetros [26]. Esta red usa solo filtros de 3x3, lo que equivale a una disminución en el número de parámetros con respecto a la arquitectura AlexNet. Esta arquitectura se encuentra reflejada en la Figura 8 (c).



(a) Arquitectura AlexNet



(b) Arquitectura GoogLeNet



(c) Arquitectura VGGNet

Figura 8: Tipos de Arquitecturas

6.Herramientas

Python:

Python es uno de los lenguajes más utilizados en Machine Learning. Es un lenguaje de programación interpretado y multiparadigma. Hemos escogido Python en vez de otros lenguajes similares, como R o Matlab, porque es un lenguaje de propósito general y su uso está mucho más extendido. Además, soporta orientación a objetos, programación imperativa y también programación funcional.

Dentro de python existen muchas librerías relacionadas con la ciencia de datos que hemos utilizado para este trabajo. La más notable es numpy que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca matemática de alto nivel para operar con esos vectores. De hecho, esta herramienta resulta de gran utilidad cuando se trabaja con Redes Neuronales, que operan con ingentes cantidades de matrices y datos. Scipy que ataca a problemas dentro de la computación científica y el análisis de datos. Y matplolib para generar gráficos a partir de datos contenidos en arrays.

Keras:

Keras es una biblioteca de código abierto de Redes Neuronales escrita para Python. Está especialmente diseñada para facilitar una rápida experimentación en Deep Learning. Permite crear prototipos rápidos de redes profundas y llevar a cabo su entrenamiento de forma cómoda y rápida.

Esta librería, ideada por Françoise Chollet en 2015, fue creada para actuar como interfaz en lugar de ser un framework de machine learning. Ofrece un conjunto de abstracciones más intuitivas y de alto nivel haciendo más sencillo el desarrollo de modelos de aprendizaje profundo independientemente del backend computacional utilizado. Asimismo, se encuentra muy integrada con otras librerías como tensorflow.

Tensorflow:

TensorFlow es una biblioteca desarrollada por Google de código abierto para aprendizaje automático. Esta librería de computación matemática ejecuta de forma rápida y eficiente gráficos de flujo. Estos gráficos están formados

por operaciones matemáticas representadas sobre nudos y cuyas entradas y salidas son vectores multidimensionales de datos(tensores).

Google Colab:

Colaboratory de Google es un entorno gratuito que no requiere configuración y se ejecuta completamente en la nube. Este entorno interactivo nos permite escribir y ejecutar código desde el navegador, pudiendo combinarlo con texto, imágenes, etc (Jupyter Notebook).

La gran ventaja de trabajar en este entorno es la posibilidad de ejecutar código en GPU potentes que ofrece Google (el código se ejecuta en "la nube"). Esto de vital importancia, pues se traduce en un ahorro más que considerable de tiempo.

Librería scikit-posthocs:

Es un paquete de Python que proporciona pruebas post hoc para comparaciones múltiples por pares, generalmente se realizan en análisis de datos estadísticos para evaluar las diferencias entre los niveles de grupo, para ver si se ha obtenido un resultado estadísticamente significativo.

7.Construcción de la red

A continuación procedemos a la construcción de la red neuronal convolucional. Para ello los parámetros a considerar serán los siguientes:

- **Función de activación:** En Keras tenemos, entre otras, las funciones de activación vistas en la sección de Aprendizaje Profundo. Pero para nuestro caso haremos uso de la ReLU, ya que es la más utilizada en las capas ocultas, porque evita el problema del gradiente evanescente de la tangente y sigmoide [27] en estas capas.

- **Optimizador:** Para que la red neuronal aprenda se necesita un método de optimización [28]. En Keras tenemos diferentes optimizadores implementados, pero nosotros para nuestra red usaremos el optimizador SGD que es el descenso estocástico del gradiente y el Adam que es un descenso de gradiente estocástico que se basa en la estimación adaptativa de los momentos de primer y segundo orden.

  Los optimizadores tienen un parámetro clave, *learning rate* o ratio de aprendizaje (α). Mediante este parámetro vamos a darle una determinada velocidad al aprendizaje de manera que se mueva más rápido o más lento hacia la mejor solución.

- **Función de coste:** Las funciones de coste son utilizadas por el optimizador para calcular el gradiente y poder buscar la mejor solución. Son funciones en las que buscamos siempre mínimos, de manera que, si queremos maximizar una función, tendremos que usar como coste su inversa.

  En Keras se encuentran una gran diversidad de funciones. Sin embargo nosotros utilizaremos una de las más típicas que se emplean para la clasificación, es la función Categorical crossentropy, utilizada para la clasificación multiclase. La podemos ver definida de la siguiente manera:

$$Loss(y, h) = -\sum_{c=1}^{c} y_c \cdot \log(h_c) \tag{5}$$

- **Métodos de regularización:** El optimizador puede provocar situaciones en las que el modelo se ajusta demasiado a datos de entrenamiento suscitando un empeoramiento general de los resultados con

datos fuera del conjunto. En este caso decimos que el modelo tiene sobreaprendizaje, u overfitting, el cual se le asocia a un sesgo, o bias, notable en el modelo.

Nosotros usaremos la técnica Batch Normalization a la hora de entrenar el modelo, ya que es mucho más rápido ir por bloques. Esta técnica consiste en hacer la normalización estadística para los datos de cada batch.

- **Capas:** Como capa final utilizaremos un perceptrón multicapa.

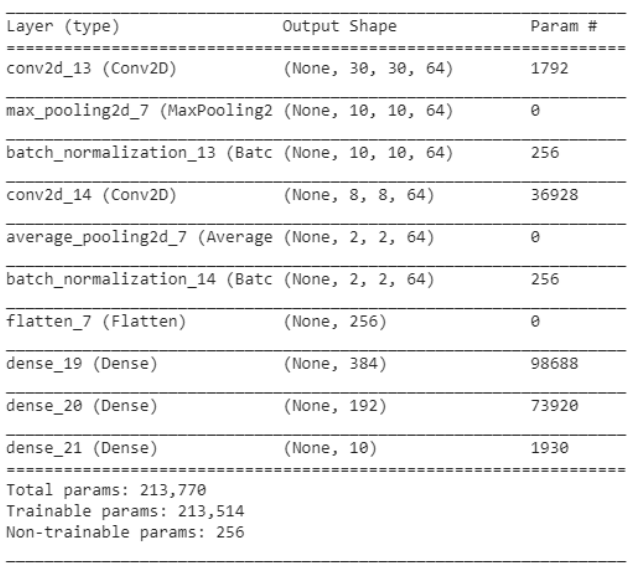
En nuestro caso hemos decidido que los parámetros a utilizar van a ser los siguientes:

- Learning rate = 0.01.
- Batch size = 32.
- Epoch = 10.

Una vez definidos los parámetros con los que vamos a trabajar procedemos a crear la arquitectura que tendrá la siguiente forma:

- Convolución 1
- Función de activación: ReLU
- Capa de Pooling
- BatchNormalization 1
- Convolución 2
- Función de activación: ReLU
- Capa de Pooling
- BatchNormalization 2
- Flatten
- Fully connected 1 (Dense con ReLU)
- Fully connected 2 (Dense con ReLU)
- Dense (con softmax)

En la Figura 9 se muestra la arquitectura de una de las configuraciones de nuestro modelo convolucional en Keras:



Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_7 (MaxPooling2	(None, 10, 10, 64)	0
batch_normalization_13 (Batc	(None, 10, 10, 64)	256
conv2d_14 (Conv2D)	(None, 8, 8, 64)	36928
average_pooling2d_7 (Average	(None, 2, 2, 64)	0
batch_normalization_14 (Batc	(None, 2, 2, 64)	256
flatten_7 (Flatten)	(None, 256)	0
dense_19 (Dense)	(None, 384)	98688
dense_20 (Dense)	(None, 192)	73920
dense_21 (Dense)	(None, 10)	1930

Total params: 213,770
Trainable params: 213,514
Non-trainable params: 256

Figura 9: Arquitectura del modelo con configuración: Optimizador:SGD, Capa pooling 1:Max, Tamaño capa pooling 1:3x3, Capa pooling 2:Avg, Tamaño capa pooling 2:4x4

De este modo, podremos decidir el tamaño de las capas de convolución y las capas de pooling. La función de activación que se va a necesitar y el tipo de capa de pooling. La capa final tendrá tantas neuronas como salidas necesite el modelo.

En nuestro caso vamos a construir la red con dos capas de convolución. Esta es una red pequeña ya que, vamos a necesitar ejecutarla muchas veces y así podemos sostener varias ejecuciones en paralelo. Además, a causa de su reducido tamaño, el efecto de los parámetros a estudiar se van a notar más.

Ambas capas tienen parámetros similares como el número de filtros (64), el tamaño del kernel (3x3) y la función de activación (ReLU). Las capas de pooling pueden trabajar tanto con el máximo (MaxPooling) como con la media (AveragePooling) y el tamaño de esta capa puede tomar los valores de 2, 3 o 4. Por último la primera capa de fully connected trabajará con 384 neuronas y la segunda capa trabajará con 192 neuronas. Finalmente, la

capa Dense del final, es decir, la capa de salida tendrá 10 neuronas correspondientes a las 10 clases que contiene el dataset. La función de activación softmax, permite entregar una salida que asigna un porcentaje de probabilidad de clasificación de la imagen de entrada.

Una vez definidos los parámetros de configuración de nuestra red necesitaremos realizar el BatchNormalization después de cada capa de pooling para mejorar la velocidad, el rendimiento y la estabilidad de nuestra red. Además, añadiremos una capa de flattening al finalizar la segunda capa de convolución para convertir la salida de la capa anterior en la entrada de un perceptrón multicapa. Una vez construida la red procedemos a entrenarla, para ello vamos a usar dos optimizadores, Adam y SGD.

8.Experimentación

Para comprobar el efecto en el rendimiento de cada componente de la red, hemos probado un total de 75 combinaciones con distintos parámetros de la misma red sobre el dataset CIFAR 10, consta de 60000 imágenes en color de 32x32 clases, con 6000 imágenes por clase. Este conjunto se compone de 50000 imágenes de entrenamiento y 10000 imágenes de prueba [29].

Como preprocesamiento de los datos se realiza una estandarización de las imágenes.

Los parámetros que hemos probado han sido:

- Tamaño ventana de convolución: 2x2 y 3x3.
- Tamaño ventana de pooling: 2x2, 3x3 y 4x4.
- Diferentes funciones para la reducción de imagen (media y máximo).
- Batch size: 32.
- Learning rate: 0.01.
- Diferentes optimizadores para la función de coste (Adam y SGD).

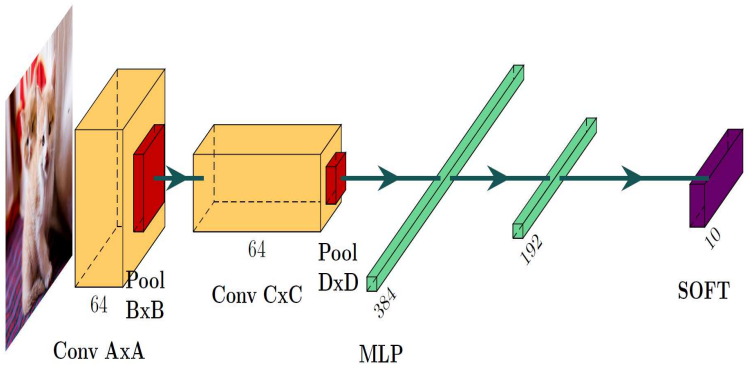


Figura 10: Esquema de la arquitectura empleada

En la Figura 10, vemos el esquema de la arquitectura empleada para nuestra experimentación, donde las letras mayúsculas representan el tamaño de las ventanas, ya que hemos cambiado dicho tamaño para la realización del estudio.

Las pruebas consistirán en realizar una comparativa entre las diferentes configuraciones para así poder ser capaces de determinar cuál de ellas obtiene mejores resultados. También será interesante comprobar la coherencia entre ellas, pues, suponiendo que todas funcionaran correctamente, entonces los resultados deberían ser razonablemente similares. Es decir, se miran primero entre las ejecuciones de la propia arquitectura, para comparar mas tarde con las poblaciones de las otras.

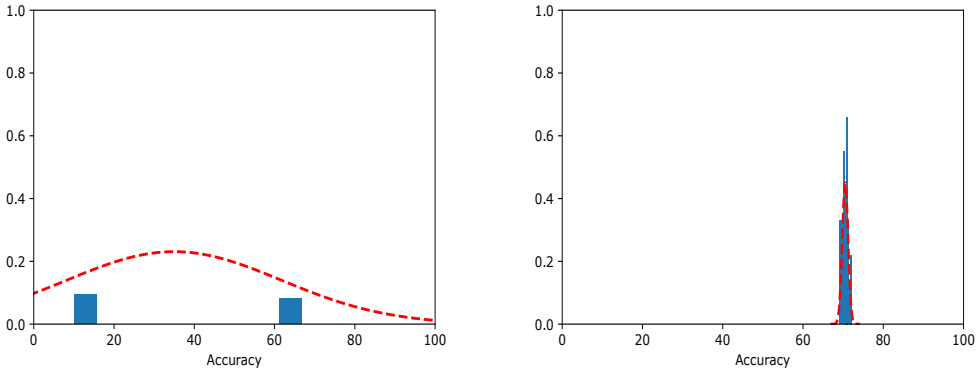
Las pruebas se han realizado en el entorno Google Colab utilizando los frameworks Keras como librería de alto nivel corriendo sobre TensorFlow como librería de más bajo nivel.

Para cada configuración, repetiremos el proceso 30 veces con el objetivo de evaluar mejor el modelo. Al hacerlo en Google Colab cada ejecución nos lleva unos 40 minutos.

De los 75 resultados obtenidos se van a seleccionar los 5 que mejor y peor media de accuracy tengan, para compararlos entre sí y compararlos con otras configuraciones con características similares.

8.1.Estabilidad en cada configuración

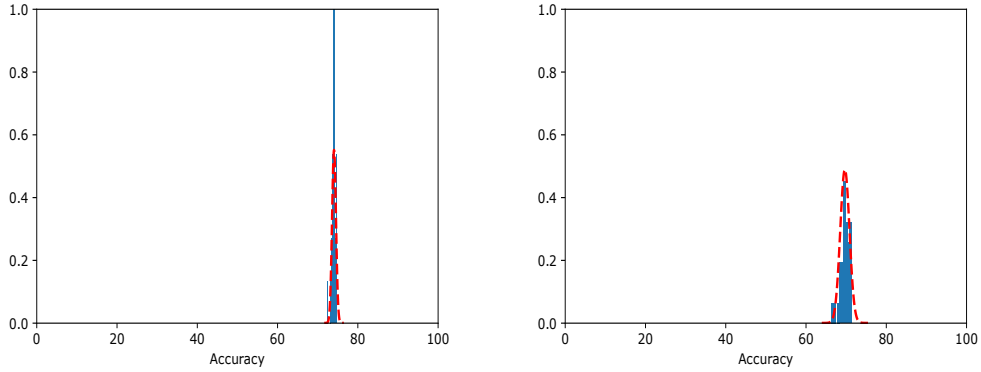
Dadas la mismas configuraciones para las ventanas de convolución y las ventanas de pooling, pero usando diferente optimizador se puede observar en las Figuras 11 y 12 que los resultados con el optimizador SGD son más estables que los del optimizador Adam.



(a) Optimizador: Adam

(b) Optimizador: SGD

Figura 11: Histogramas de la configuración: Tamaño de convolución: 3x3, Tamaño de pooling: 2x2, Función pooling: Máximo



(a) Optimizador: SGD

(b) Optimizador: Adam

Figura 12: Histogramas de la configuración: Tamaño de convolución: 3x3, Tamaño de pooling 1: 3x3, Tamaño de pooling 2: 4x4, Función pooling 1: Máximo, Función pooling 2: Media

En la Figura 11, la configuración del optimizador Adam corresponde a la peor configuración obtenida, en cambio la Figura 12, la configuración del optimizador SGD corresponde a la mejor de todas ellas.

De la Figura 11, se puede deducir la gran diferencia entre ambas configuraciones, ya que, aunque se tengan los mismos parámetros a diferencia del optimizador, la media de accuracy obtenidas son muy diferentes, siendo en el caso del optimizador Adam de 34.96 y del optimizador SGD de 70.51. También cabe destacar, que en los resultados obtenidos en el optimizador Adam la desviación típica es elevada, 26.71, frente a la del optimizador SGD, que es de 0.73.

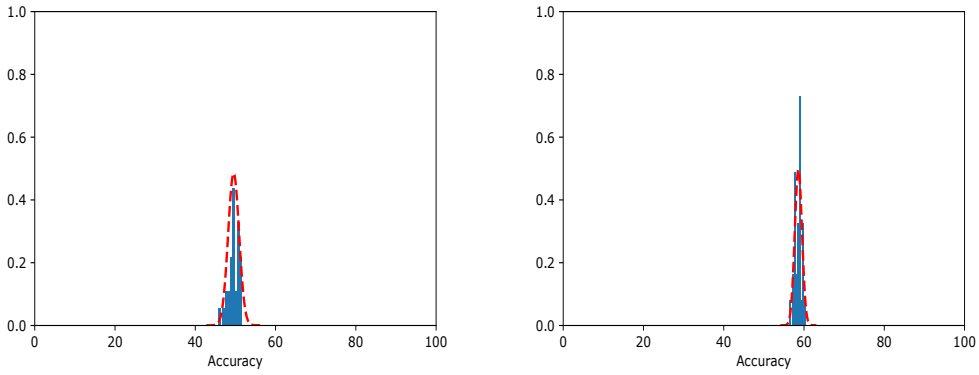
Del mismo modo, analizamos la Figura 12, obteniendo resultados similares, ya que no hay tanta diferencia entre la media de acuraccy y la desviación típica obtenida, siendo en Adam la media 69.68 y desviación 1.15, y en SGD la media 74.07 y desviación 0.49.

De ambas figuras, se puede deducir la gran diferencia existente entre la mejor y peor configuración, debido a la tasa de acierto obtenida en cada una de ellas.

8.2.Cambios en la ventana de pooling (tamaño y función)

Dadas la mismas configuraciones para las ventanas de convolución, pero usando diferentes optimizadores y diferentes ventanas de pooling, se puede observar en las Figuras 13 y 14 que los resultados con el optimizador SGD siguen siendo más estables que los del optimizador Adam.

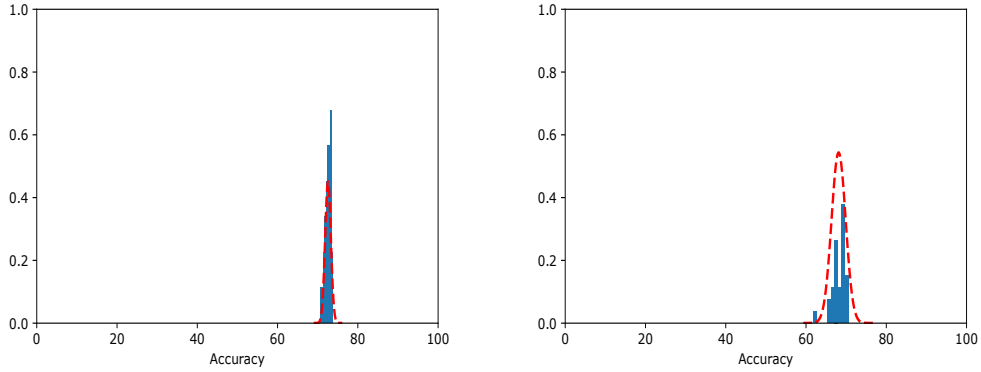
En este caso, estudiaremos la mejor de las 5 peores configuraciones y la peor de las 5 mejores configuraciones.



(a) Optimizador: Adam

(b) Optimizador: SGD

Figura 13: Histogramas de la configuración: Tamaño de convolución: 3x3, Tamaño de pooling 1: 4x4, Tamaño de pooling 2: 3x3, Función pooling 1: Media, Función pooling 2: Máximo



(a) Optimizador: SGD

(b) Optimizador: Adam

Figura 14: Histogramas de la configuración: Tamaño de convolución: 3x3, Tamaño de pooling 1: 3x3, Tamaño de pooling 2: 2x2, Función pooling: Máximo

En la Figura 13, la configuración del optimizador Adam corresponde a la mejor de las peores de las configuraciones obtenidas, en cambio la Figura 14, la configuración del optimizador SGD corresponde a la peor de las mejores de todas ellas.

De la Figura 13, se puede deducir la escasa diferencia entre ambas configuraciones, ya que, aunque se tengan los mismos parámetros a diferencia del optimizador, la media de accuracy obtenidas son muy diferentes, siendo en el caso del optimizador Adam de 49.55 y del optimizador SGD de 58.60. También cabe destacar, que en los resultados obtenidos en el optimizador Adam la desviación típica es un poco mayor, 1.36, que la del optimizador SGD, que es de 0.90.

Del mismo modo, analizamos la Figura 14, obteniendo resultados similares, ya que no hay tanta diferencia entre la media de acuraccy y la desviación típica obtenida, siendo en Adam la media 68.12 y desviación 1.76, y en SGD la media 72.57 y desviación 0.71.

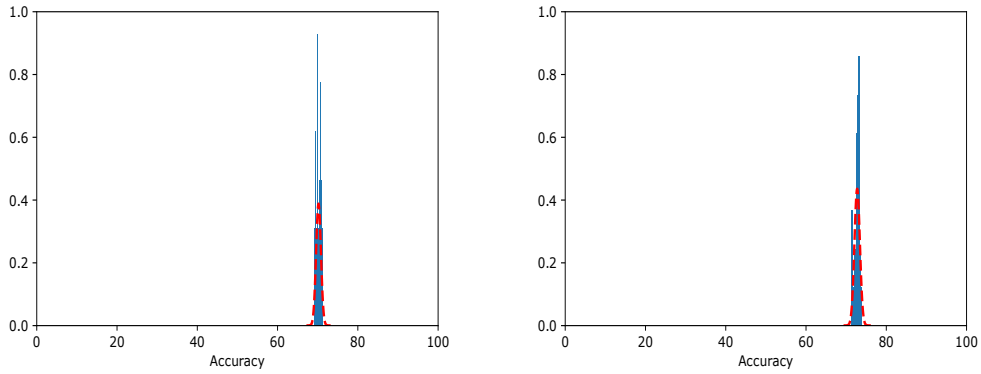
De ambas figuras, podemos deducir con relación a la función de pooling, que al tener la misma función en ambas capas los resultados obtenidos de la

tasa de acierto varian menos que teniendo diferentes funciones de pooling. Con respecto al tamaño de las ventanas observamos que funciona mejor la segunda combinación de tamaños.

De estas 4 configuraciones, se puede ver que en la Figura 14 la configuración con el optimizador SGD es la que mejor se puede adaptar a una distribución normal.

8.3.Cambios en la ventana de convolución

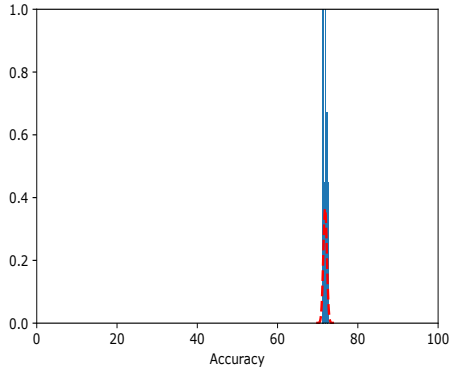
Dada la mejor configuración de las combinaciones probadas, analizamos esa misma configuración pero con diferente tamaño de ventana de convolución.



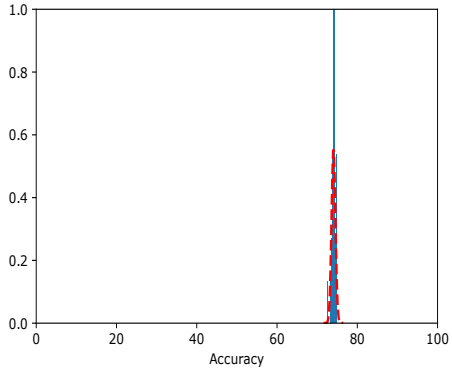
(a) Tamaño de convolución 1: 2x2, Tamaño de convolución 2: 2x2

(b) Tamaño de convolución 1: 2x2, Tamaño de convolución 2: 3x3

Figura 15: Histogramas de la configuración: Optimizador: SGD, Tamaño de pooling 1: 3x3, Función pooling 1: Máximo, Tamaño de pooling 2: 4x4,Función pooling 2: Media



(a) Tamaño de convolución 1: 3x3, Tamaño de convolución 2: 2x2



(b) Tamaño de convolución 1: 3x3, Tamaño de convolución 2: 3x3

Figura 16: Histogramas de la configuración: Optimizador: SGD, Tamaño de pooling 1: 3x3, Función pooling 1: Máximo, Tamaño de pooling 2: 4x4, Función pooling 2: Media

De las Figuras 15 y 16, podemos observar que la configuración 16 (b) corresponde a la mejor configuración que hemos obtenido.

De ambas figuras, podemos deducir con relación al tamaño de la ventana de convolución que el mejor tamaño para las dos ventanas de convolución es de 3x3.

8.4.Diferencias estadísticas entre configuraciones

En la Tabla 1 mostramos los 5 mejores y peores resultados obtenidos durante la ejecución de las diferentes configuraciones. En el Anexo I nos encontramos la tabla completa de los resultados obtenidos.

Optimizador	**Pooling 1**	**Tamaño 1**	**Pooling 2**	**Tamaño 2**	**Media**	**Desviación típica**
Adam	Max	2x2	Max	2x2	34.96	26.71
Adam	Avg	2x2	Avg	2x2	42.05	24.41
Adam	Avg	2x2	Max	2x2	42.35	24.66
Adam	Max	2x2	Avg	2x2	42.95	25.10
Adam	Avg	4x4	Max	3x3	49.55	1.36
SGD	Max	3x3	Max	2x2	72.57	0.71
SGD	Max	2x2	Avg	3x3	73.00	0.51
SGD	Max	3x3	Avg	2x2	73.60	0.39
SGD	Max	2x2	Avg	4x4	73.74	0.40
SGD	Max	3x3	Avg	4x4	74.07	0.49

Tabla 1: Resultados de las 5 mejores y peores configuraciones encontradas

Como podemos observar, los resultados al evaluar cada modelo tiene una media de tasa de acierto elevada, es decir, podemos intuir como de bien va a clasificar el modelo sobre el conjunto de datos que nunca ha visto anteriormente. De ellos se puede resaltar que los peores resultados se obtienen de la configuración que usa el optimizador Adam, y más concretamente el peor de todos ellos emplea en la primera capa de pooling la media y en la segunda capa de pooling el máximo.

En la Tabla 1 podemos observar que la diferencia obtenida entre la mejor y la peor configuración es de 39.11 puntos de acierto.

Una vez probadas todas las variaciones de las configuraciones, se les ha realizado un test de normalidad a cada una de las soluciones obtenidas, quedándonos entre ellos con los 5 mejores resultados, los cuales son los siguientes:

- Configuración 1 con media 72.57

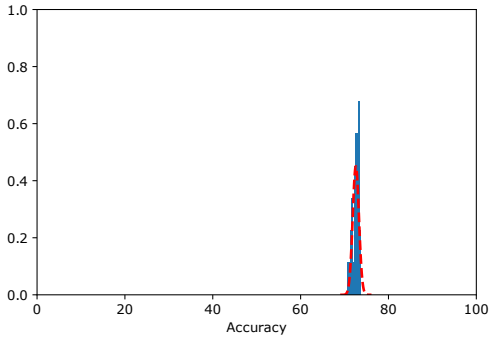


Figura 17: Optimizador: SGD, Capa convolución: 3x3, Capa pooling 1: Máximo, Tamaño capa pooling 1: 3x3, Capa pooling 2: Máximo, Tamaño capa pooling 2: 2x2

- Configuración 2 con media 73.00

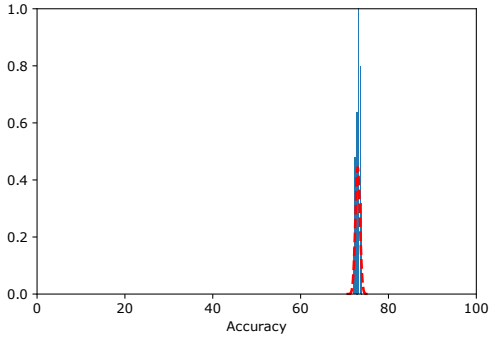


Figura 18: Optimizador: SGD, Capa convolución: 3x3, Capa pooling 1: Máximo, Tamaño capa pooling 1: 2x2, Capa pooling 2: Media, Tamaño capa pooling 2: 3x3

- Configuración 3 con media 73.60

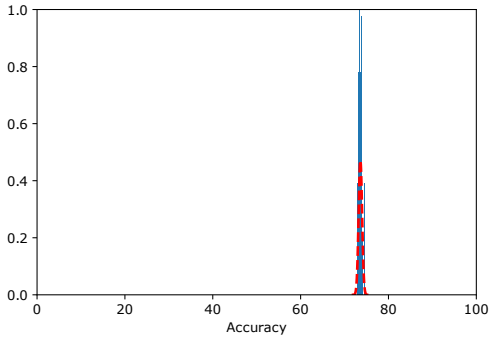


Figura 19: Optimizador: SGD, Capa convolución: 3x3, Capa pooling 1: Máximo, Tamaño capa pooling 1: 3x3, Capa pooling 2: Media, Tamaño capa pooling 2: 2x2

- Configuración 4 con media 73.74

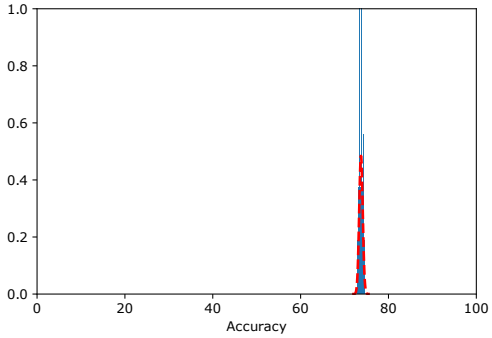


Figura 20: Optimizador: SGD, Capa convolución: 3x3, Capa pooling 1: Máximo, Tamaño capa pooling 1: 2x2, Capa pooling 2: Media, Tamaño capa pooling 2: 4x4

- Configuración 5 con media 74.07

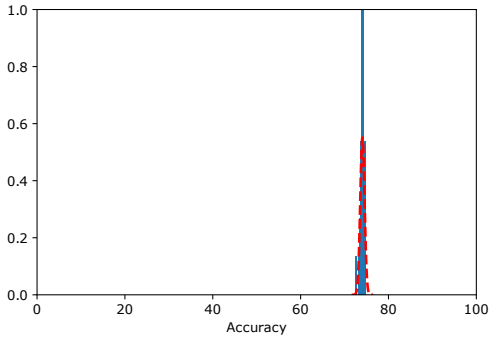


Figura 21: Optimizador: SGD, Capa convolución: 3x3, Capa pooling 1: Máximo, Tamaño capa pooling 1: 3x3, Capa pooling 2: Media, Tamaño capa pooling 2: 4x4

Estudio de diferencias estadísticas significativas

A estas cinco mejores combinaciones les hemos realizado el test estadístico multipoblacional de Friedman, este test lo usamos para comparar si hay diferencias estadísticas entre varias poblaciones. Si el resultado de este test nos da un p-valor menor que 0.05 usaremos el post-hoc de Nemenyi, debido a que nos encontramos frente a poblaciones que no siguen una distribución normal, para saber cuáles de ellas son distintas de otras.

Como el resultado del primer test nos ha dado un p-valor mayor que 0.05, no hemos realizado el segundo test.

Análisis de los resultados

Los resultados que hemos recogido nos permiten evaluar el modelo con distintas configuraciones, y esto es clave en nuestro afán por descubrir cuál de ellas funciona mejor en la práctica.

Para evaluarlos, hemos recogido las medias obtenidas de cada uno de ellos. Se evaluarán dependiendo del tipo de optimizador empleado, y las diferentes combinaciones que puede tomar la capa de pooling.

Una vez expuesto el sistema de evaluación, podemos pasar por fin a comentar sus resultados. Estos resultados se exponen en la siguientes tablas. En ellas se pueden ver por cada optimizador y combinación, cuál es la media del accuracy obtenido para cada una de ellas.

Optimizador	**AvgAvg**	**AvgMax**	**MaxMax**	**MaxAvg**
Adam	57.69	57.76	61.23	62.21
SGD	68.21	66.54	69.02	70.55

Tabla 2: Resultados independientes del tamaño de la ventana de pooling

En la Tabla 2 podemos ver reflejadas qué combinaciones funcionan mejor y peor (en términos relativos con respecto al resto de combinaciones). En esta tabla se encuentran reflejadas por optimizador y combinación de funciones de las capas de pooling, las medias obtenidas por cada una de ellas.

Optimizador	**AvgAvg**	**AvgMax**	**MaxMax**	**MaxAvg**
Adam	42.05	42.35	34.96	42.95
SGD	69.40	68.00	70.51	71.87

Tabla 3: Resultados con tamaño 3x3 en la ventana de convolución y 2x2 en la ventana de pooling

Optimizador	**AvgAvg**	**AvgMax**	**MaxMax**	**MaxAvg**
Adam	55.02	53.86	60.11	60.28
SGD	64.24	61.99	64.38	66.70

Tabla 4: Resultados con tamaño 3x3 en la ventana de convolución y 4x4 en la ventana de pooling

En la Tabla 3 y 4 observamos diferencias significativas de trabajar con ventanas de pooling pequeñas o grandes. En el caso del optimizador Adam, se obtiene una mejor tasa de acierto empleando un tamaño grande en la ventana de pooling. En cambio, el optimizador SGD obtiene mejores resultados cuando trabaja con ventanas de pooling pequeñas.

Como ya habíamos comentado en el apartado anterior y queda más que evidente en las tablas, el modelo que emplea el optimizador Adam es el que peores resultados obtiene. El mejor de ellos lo obtiene con la combinación de máximo y media, pero queda muy lejos de los obtenidos con el optimizador SGD.

La técnica ganadora de las comparativas resulta ser SGD usando máximo y media en sus respectivas capas de pooling. Esta es la que mejor media de accuracy obtiene.

Después de realizar todas las pruebas, al no obtener un p-valor menor que 0.05, nos fijamos en la media de cada uno de los accuracy de las cinco mejores configuraciones, donde observamos que la mejor media es 74.07 y se encuentra con la configuración:

- Optimizador: SGD
- Primera capa de pooling: Máximo
- Segunda capa de pooling: Media
- Tamaño primera capa de pooling: 3x3
- Tamaño segunda capa de pooling: 4x4

Podemos deducir que la configuración anterior es la mejor respecto a las otras probadas.

9.Conclusiones

En este trabajo hemos estudiado el efecto en el rendimiento de diferentes parámetros en una red convolucional. Para conseguirlo hemos ejecutado distintas configuraciones de la misma arquitectura de red, cambiando el optimizador, el tamaño de ventana de convolución, de pooling y la función de esta. Para obtener fiabilidad estadística en nuestro resultado hemos ejecutado 30 veces cada red siguiendo una iniciación aleatoria de los pesos. Tras realizar todas las ejecuciones, no se han encontrado resultados significativos, es decir, al trabajar con las 5 configuraciones que mejor tasa de acierto tienen no hemos encontrado diferencias reveladoras entre ellas. Aunque, cabe destacar que, entre la mejor y la peor configuración encontradas hay una diferencia de casi 40 puntos de acierto.

Particularmente, hay que tener en cuenta los efectos de los diferentes parámetros empleados, ya que, según cuáles escojamos podemos llegar a obtener un resultado muy bueno o muy malo. Por ese motivo, cabe destacar que nuestros resultados apuntan a que el optimizador puede ser decisivo en el rendimiento final del sistema, por esa razón, hemos decidido trabajar con Adam y SGD, ya que vimos que RMSprop y Adadelta no afectan tanto.

Finalmente, de todos los parámetros estudiados observamos que el que más afecta al rendimiento es el optimizador empleado. Una vez visto los resultados dados por los dos optimizadores, el siguiente parámetro que más afecta es el tamaño de la ventana de pooling, ya que del mejor al peor resultado le saca 26.30 puntos de acierto en el caso del optimizador Adam y 11.68 puntos en el caso del SGD. Por otra parte, los dos parámetros que menos afectan al rendimiento son el tamaño de la ventana de convolución y el filtro empleado en la capa de pooling, debido a que, en ningún caso del mejor al peor resultado hay una diferencia mayor a 5 puntos de acierto.

Bibliografía

[1] *Inteligencia Artificial: un enfoque moderno, segunda edición, de Stuart Russell y Peter Norving*

[2] *Multiple layer perceptron training using genetic algorithms, de Udo Seiffert, University of South Australia, Adelaide Knowledge-Based Intelligent Engineering Systems Centre (KES), ESANN'2001 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 159-164*

[3] *VIVAS, Hevert; MARTINEZ, Héctor Jairo and PEREZ, Rosana. Método secante estructurado para el entrenamiento del perceptrón multicapa. rev. cienc. [online]. 2014, vol.18, n.2, pp.131-150. ISSN 0121-1935.*

[4] *Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533536 (1986).*

[5] *Los robots y la Inteligencia Artificial: nuevos retos del periodismo, de Idoia Salazar*

[6] *Rina Dechter (1986). Learning while searching in constraint-satisfaction problems. University of California, Computer Science Department, Cognitive Systems Laboratory.*

[7] *One weird trick for parallelizing convolutional neural networks, de Alex Krizhevsky, 2014*

[8] *Chen, D., Liu, S., Kingsbury, P. et al. Deep learning and alternative learning strategies for retrospective real-world clinical data. npj Digit. Med. 2, 43 (2019).*

[9] *Li Deng and Dong Yu. 2014. Deep Learning: Methods and Applications. Found. Trends Signal Process. 7, 34 (June 2014), 197387.*

[10] *Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M. et al. Deep learning applications and challenges in big data analytics. Journal of Big Data 2, 1 (2015).*

[11] *A. Lucas, M. Iliadis, R. Molina and A. K. Katsaggelos, Üsing Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods,ïn IEEE Signal Processing Magazine, vol. 35, no. 1, pp. 20-36, Jan. 2018.*

[12] *Costajussà, M., & Fonollosa, J. (2017). DeepVoice: Tecnologías de Aprendizaje Profundo aplicadas al Procesado de Voz y Audio. Procesamiento Del Lenguaje Natural, 59, 117-120.*

[13] *McCarthy J. (1989) Artificial Intelligence, Logic and Formalizing Common Sense. In: Thomason R.H. (eds) Philosophical Logic and Artificial Intelligence. Springer, Dordrecht, ISBN: 978-94-009-2448-2*

[14] *Lessons from a Restricted Turing Test,Stuart M. Shieber,1994*

[15] *Artificial Neural Networks and Machine Learning – ICANN 2012: 22nd International Conference on Artificial Neural Networks, Lausanne, Switzerland, September 1114, 2012, Proceedings, Part II, de Alessandro Villa, Wlodzislaw Duch, Péter Érdi, Francesco Masulli y Günther Palm.*

[16] *Rojas R. (1996) Weighted NetworksThe Perceptron. In: Neural Networks. Springer, Berlin, Heidelberg., ISBN 978-3-642-61068-4*

[17] *Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6),386408.*

[18] *Vivas, Hevert, Martínez, Héctor Jairo, & Pérez, Rosana. (2014). Método secante estructurado para el entrenamiento del perceptrón multicapa. Revista de Ciencias, 18(2), 131-150.*

[19] *B. Moons, R. Uytterhoeven, W. Dehaene y M. Verhelst, "14.5 Envision: un procesador de red neuronal convolucional escalable de precisión de voltaje de 0,26 a 10TOPS / W paralelo paralelo con precisión de frecuencia dinámica escalable en FDSOI de 28nm", 2017 IEEE International*

Solid-State Circuits Conference (ISSCC) , San Francisco, CA, 2017, pp. 246247, doi: 10.1109 / ISSCC.2017.7870353.

[20] *Detección automática de especies bentónicas en fondos profundos mediante imágenes obtenidas con vehículos remotos y técnicas de deep-learning, de Celia Díaz Cuesta, 2018, Universidad de Cantabria*

[21] *Deep Learning aplicado al análisis de soldadura mediante espectroscopía óptica de plasmas, de César Abascal Gutiérrez, 2017, Universidad de Cantabria*

[22] *Ejemplos de aplicación de pooling*
`https://cdn-images-1.medium.com/max/750/1*KQIEqhxzICU7thjaQBfPBQ.png`

[23] *S. Hershey et al., ÇNN architectures for large-scale audio classification,"2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 131-135, doi: 10.1109/ICASSP.2017.7952132.*

[24] *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches, Md Zahangir Alom and Tarek M. Taha and Christopher Yakopcic and Stefan Westberg and Paheding Sidike and Mst Shamima Nasrin and Brian C Van Esesn and Abdul A S. Awwal and Vijayan K. Asari,2018*

[25] *Going Deeper With Convolutions, Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9*

[26] *Deep Captioning with Multimodal Recurrent Neural Networks (mRNN), Junhua Mao and Wei Xu and Yi Yang and Jiang Wang and Zhiheng Huang and Alan Yuille, 2014*

[27] *Zou, D., Cao, Y., Zhou, D. et al. Gradient descent optimizes over-parameterized deep ReLU networks. Mach Learn 109, 467492 (2020).*

[28] *R. F. Lyon, "Neural Networks for Machine Learning", Hum. Mach. Hear., pp. 419-440, 2017.*

[29] *RC Çalik y MF Demirci, "Clasificación de imágenes Cifar10 con redes neuronales convolucionales para sistemas embebidos", 15a Conferencia in-*

ternacional IEEE / ACS 2018 sobre sistemas y aplicaciones informáticas (AICCSA) , Aqaba, 2018, pp. 1-2, doi: 10.1109 /AICCSA.2018.8612873.

[30] *Redes de Neuronas Artificiales Un Enfoque Práctico, de Pedro Isasi Viñuela e Inés M. Galvan, 2004. Madrid España.*

[31] *Las Redes Neuronales Artificiales. Fundamentos teóricos y aplicaciones prácticas, de Raquel Flórez López y José Miguel Fernández Fernández, 2008, ISBN: 978-84-9745-246-5*

[32] *Machine Learning from Theory to Algorithms: An Overview, Jafar Alzubi and Anand Nayyar and Akshi Kumar, 2018 J. Phys.: Conf. Ser. 1142 012012*

Anexo I: Tabla de resultados

Configuración	Media	Desviación típica	Error
ConfAdam_AA_2x2	42,05	24,41	0,81
ConfAdam_AA_2x3	63,97	1,14	0,04
ConfAdam_AA_2x4	64,71	1,43	0,05
ConfAdam_AA_3x2	62,86	1,70	0,06
ConfAdam_AA_3x3	59,00	1,33	0,04
ConfAdam_AA_3x4	64,90	1,27	0,04
ConfAdam_AA_4x2	56,81	1,91	0,06
ConfAdam_AA_4x3	49,88	1,29	0,04
ConfAdam_AA_4x4	55,02	1,53	0,05
ConfAdam_AM_2x2	42,35	24,66	0,82
ConfAdam_AM_2x3	64,68	1,11	0,04
ConfAdam_AM_2x4	66,38	0,91	0,03
ConfAdam_AM_3x2	63,03	1,24	0,04
ConfAdam_AM_3x3	59,66	1,57	0,05
ConfAdam_AM_3x4	63,79	1,22	0,04
ConfAdam_AM_4x2	56,58	1,64	0,05
ConfAdam_AM_4x3	49,55	1,36	0,05
ConfAdam_AM_4x4	53,86	1,74	0,06
ConfAdam_MA_2x2	42,95	25,10	0,84
ConfAdam_MA_2x3	66,53	1,22	0,04
ConfAdam_MA_2x4	68,16	0,88	0,03
ConfAdam_MA_3x2	68,05	1,31	0,04
ConfAdam_MA_3x3	64,72	1,08	0,04
ConfAdam_MA_3x4	69,68	1,15	0,04
ConfAdam_MA_4x2	63,96	1,11	0,04
ConfAdam_MA_4x3	55,55	1,03	0,03
ConfAdam_MA_4x4	60,28	1,02	0,03
ConfAdam_MM_2x2	34,96	26,71	0,89
ConfAdam_MM_2x3	66,77	1,40	0,05
ConfAdam_MM_2x4	68,29	1,17	0,04
ConfAdam_MM_3x2	68,12	1,76	0,06
ConfAdam_MM_3x3	64,28	0,95	0,03
ConfAdam_MM_3x4	68,81	1,01	0,03
ConfAdam_MM_4x2	63,95	1,02	0,03

Configuración	Media	Desviación típica	Error
ConfAdam_MM_4x3	55,79	0,87	0,03
ConfAdam_MM_4x4	60,11	0,90	0,03
ConfSGD_AA_2x2	69,40	0,76	0,03
ConfSGD_AA_2x3	70,90	0,85	0,03
ConfSGD_AA_2x4	71,81	0,78	0,03
ConfSGD_AA_3x2	70,98	0,96	0,03
ConfSGD_AA_3x3	67,89	1,22	0,04
ConfSGD_AA_3x4	72,09	0,90	0,03
ConfSGD_AA_4x2	66,83	0,54	0,02
ConfSGD_AA_4x3	59,73	1,17	0,04
ConfSGD_AA_4x4	64,24	0,73	0,02
ConfSGD_AM_2x2	68,00	0,62	0,02
ConfSGD_AM_2x3	69,31	0,48	0,02
ConfSGD_AM_2x4	70,08	0,57	0,02
ConfSGD_AM_3x2	69,69	0,68	0,02
ConfSGD_AM_3x3	66,12	0,76	0,03
ConfSGD_AM_3x4	69,80	0,8	0,03
ConfSGD_AM_4x2	65,30	0,84	0,03
ConfSGD_AM_4x3	58,60	0,9	0,03
ConfSGD_AM_4x4	61,99	0,86	0,03
ConfSGD_MA_2x2	71,87	0,51	0,02
ConfSGD_MA_2x3	73,00	0,51	0,02
ConfSGD_MA_2x4	73,74	0,4	0,01
ConfSGD_MA_3x2	73,60	0,39	0,01
ConfSGD_MA_3x3	70,37	0,61	0,02
ConfSGD_MA_3x4	74,07	0,49	0,02
ConfSGD_MA_4x2	69,22	0,55	0,02
ConfSGD_MA_4x3	62,36	0,44	0,01
ConfSGD_MA_4x4	66,70	0,59	0,02
ConfSGD_MM_2x2	70,51	0,73	0,02
ConfSGD_MM_2x3	71,87	0,57	0,02
ConfSGD_MM_2x4	72,31	0,48	0,02
ConfSGD_MM_3x2	72,57	0,71	0,02
ConfSGD_MM_3x3	68,70	0,62	0,02
ConfSGD_MM_3x4	72,37	0,44	0,01

Configuración	**Media**	**Desviación típica**	**Error**
ConfSGD_MM_4x2	67,57	0,58	0,02
ConfSGD_MM_4x3	60,95	0,53	0,02
ConfSGD_MM_4x4	64,38	0,84	0,03

Tabla 5: Resultados de configuraciones encontradas