Universidad Internacional de La Rioja (UNIR)

ESIT

Máster Universitario en Inteligencia Artificial

Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes

Trabajo Fin de Máster

Presentado por: Moreno Díaz-Alejo, Lara

Director/a: Ruiz Iniesta, Almudena

Ciudad: Madrid

Fecha: Enero 2020

Tabla de contenido

Tabla de ilustraciones

Resumen

La clasificación de imágenes ha sido una de las principales áreas de investigación dentro del campo de visión artificial. Desde hace algo más de una década ha estado dominado por las redes neuronales convolucionales, CNN, que a pesar de los buenos resultados y grandes éxitos que han cosechado, sufren de varios inconvenientes. Para solventar estos problemas, recientemente se ha desarrollado un nuevo algoritmo de redes neuronales llamado redes de cápsulas (*CapsNet*) basado en vectores y en un nuevo algoritmo de entrenamiento llamado algoritmo de enrutamiento dinámico, de manera que podría mejorar los resultados de conocidas arquitecturas para la clasificación de imágenes como lo son las redes neuronales convolucionales. En este estudio se ha probado el rendimiento de esta nueva arquitectura, y se ha comparado con una arquitectura convolucional basada en bloques residuales. Para contrastar los resultados se han usado conjuntos de datos de distintas complejidades (MNIST, Fashion-MNIST, COIL-100 y CIFAR-10). A pesar de la buena perspectiva teórica que ofrece este nuevo modelo en la teoría, los resultados obtenidos todavía no son lo esperados en cuanto se manejan conjuntos de datos con cierta complejidad. Con MNIST y muy cerca, Fashion-MNIST sí se han mejorado los resultados, mientras que con los conjuntos de datos más complejos (COIL-100 Y CIFAR-10), los resultados no han sido los esperados.

Palabras Clave: CNN, CapsNet, Convolucionales, Redes neuronales, Cápsulas de redes.

Abstract

Image classification has been one of the main research areas within the field of artificial vision. For more than a decade it has been dominated by convolutional neural networks, CNN, although the good results and great successes they have achieve, they have the risk of several drawacks. To solve these problems, a new neural network algorithm called capsule networks (CapsNet) based on vectors, and a new training algorithm called dynamic routing algorithm has recently been developed, so that it could improve the results of known architectures for the classification of images, like convolutional neural networks. In this study we have tested the performance of this new architecture, and we have compared it with a convolutional architecture based on residual blocks. To contrast the results we have used datasets of different complexities (MNIST, Fashion-MNIST, COIL-100 and CIFAR-10). Despite the good theoretical perspective offered by this new model in theory, the results are not yet expected as soon as we handle datasets with some complexity. The results have been improved with

MNIST and almost with Fashion-MNIST, while with more complex datasets (COIL-100 and CIFAR-10) the results have not been as expected.

Keywords: Convolutional, Capsules, Network, DataSet

1. Introducción.

La clasificación de imágenes es una de las áreas dentro del campo de visión artificial que más se ha estudiado en la última década. Dicha tarea ha generado gran interés en la comunidad en general debido a sus aplicaciones actuales y futuras potenciales. Reconocimientos faciales en los móviles, coches que empiezan a ser autónomos en la conducción, retos que premian a los mejores proyectos clasificando y detectando imágenes, reconocimiento de huellas digitales, predicciones en el área de tráfico, radares que capturan matrículas y un largo etcétera. Todo esto se traduce en una gran cantidad de estudios científicos trabajando en las tareas de clasificación y reconocimiento de imágenes, detección/localización de objetos, generación de imágenes y muchas otras tareas directamente relacionadas con la visión por computador. Este campo ha estado dominado en la última década por las redes neuronales. Y en especial, la tarea de clasificación de imágenes ha estado dominada por las redes neuronales convolucionales. Estas redes han sido muy estudiadas, probadas y desarrolladas, de hecho, existen numerosas y diferentes arquitecturas diseñadas con gran éxito en la tarea de reconocimiento de imágenes.

La referencia estándar del inicio de las redes neuronales convolucionales, CNN, es de 1989, aunque antes de este año, a finales de la década de los 70 ya existió una red neuronal artificial llamada *Neocognitrón* (Fukushima, 1980) que fue la base para las redes convolucionales como ahora las conocemos. Neocognitrón (Fukushima, 1980) estaba formada por varias capas y realizaba reconocimiento de caracteres escritos a mano. Pero la primera red convolucional nació en 1998 y se llamó *LeNet-5* (LeCun et al., 1998a). Ésta era una arquitectura sencilla, con 7 capas, dedicada a reconocer números manuscritos en los cheques de algunos bancos de Estados Unidos. A partir de ese año, se incrementó considerablemente el estudio de este tipo de redes, y es en 2014 cuando de la mano de la existencia de grandes bases de datos de imágenes, y de los avances tecnológicos en computación, se convirtieron en un referente en el campo de la clasificación de imágenes gracias a sus buenos resultados.

A pesar de su reconocido y probado éxito, las redes convolucionales tienen dos grandes inconvenientes: no tienen en cuenta las jerarquías espaciales que existen entre los diferentes elementos de la imagen y la propia imagen, y sufren la falta de invarianza rotacional. Es decir, las redes convolucionales aprenden la detección de características en el plano XY de la imagen, o mapa de características de entrada, pero no en el espacio de rotación. Con los datos de prueba, el no tener en cuenta estas jerarquías espaciales, es decir sin tener en cuenta la orientación espacial entre las características de la imagen hace que se produzcan falsos positivos. Y la falta de invarianza rotacional hace que se produzcan falsos negativos, ya que la red asignaría incorrectamente el objeto a la etiqueta que no corresponde.

Las redes neuronales convolucionales obtienen unos resultados excelentes reconociendo diferentes características dentro de una imagen. Por ejemplo, aprendiendo a reconocer los elementos de una cara (dos ojos, una nariz y una boca) y deduciendo luego, que algo que contenga estos elementos, probablemente será una cara. Sin embargo, este tipo de redes realizan numerosas y complejas operaciones y uno de los inconvenientes que sufren es que con algunas de ellas se pierde información. Esta información puede ser, por ejemplo, sobre orientación y/o disposición y colocación de los elementes entre sí, y a pesar de esta pérdida de información, seguirá deduciendo que es una cara. Como se ve en la Figura 1, ambas imágenes tienen los mismos elementos, pero están colocados en distintas posiciones y con distinta orientación. Si una red neuronal, pierde información y no puede tener en cuenta las relaciones espaciales que hay entre los distintos elementos, podría clasificar la imagen de la derecha como una cara correcta, cuando no lo es.

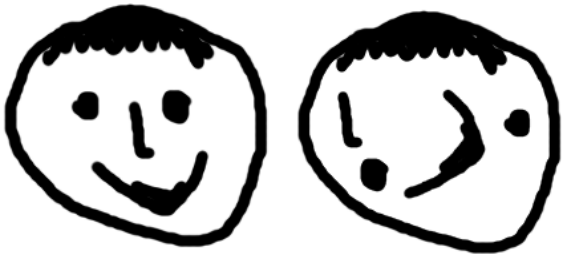

FIGURA 1 CLASIFICACIÓN CORRECTA/INCORRECTA

Fuente: Elaboración propia.

Otros problemas que también se ha demostrado que sufren las redes convolucionales muy profundas y complejas es que, llegado un punto, dejan de aprender y aparecen tres principales problemas: la degradación/explosión del gradiente (*vanishing/exploding gradient*) y la maldición de la dimensionalidad (*curse of dimensionality*) que se detallan más adelante.

Estudiando estos principales inconvenientes que sufren las redes convolucionales, Geoffrey Hinton publicó en 2011 la teoría de cápsulas de redes (G. E. Hinton et al., 2011). A Hinton, se le puede considerar como uno de los fundadores del *Deep Learning*, y ha desarrollado numerosos modelos y algoritmos que han sido y siguen siendo ampliamente estudiados y usados por toda la comunidad científica. Este nuevo modelo de red se basa en agrupar las neuronas en cápsulas, y la principal diferencia es que, así como las neuronas tienen como salida un único escalar donde guardar la información, las redes de cápsulas tienen como salida un vector, dando lugar así a poder generar y guardar mucha más información de la que puede guardar una sola neurona, con la consecuente mejora a la hora de realizar las

predicciones. A pesar de publicar el artículo sobre cápsulas de redes en 2011, no fue hasta el año 2017 cuando se publicó un método para entrenar este tipo de redes.

Debido a la novedad y al potencial que nos presentan con esta arquitectura, en este estudio se ha diseñado una red neuronal basada en redes de cápsulas y se ha probado el rendimiento de este nuevo modelo, con tres modelos basados en una arquitectura de redes convolucionales como *ResNet* (LeCun & Cortes, 2010), donde la diferencia entre cada modelo *ResNet* (LeCun & Cortes, 2010) elegido es la profundidad de la red. Se ha elegido *ResNet* (LeCun & Cortes, 2010), por ser esta arquitectura ampliamente estudiada, poder variar la profundidad de la red, y por sus buenos resultados. Se usarán los conjuntos de datos *MNIST* (LeCun & Cortes, 2010), *CIFAR-10* (Krizhevsky et al., 2009), *Fashion-Mnist* (Xiao et al., 2017) y *Coil100* (Nene et al., 1996). También, se ha aplicado la función de activación *ReLU*, *Cross-Entropy* para calcular el error y se ha entrenado la red usando el algoritmo *Batch Stochastic Gradient Descent* (LeCun et al., 1998a), y para la red de cápsulas se usará el algoritmo de enrutamiento dinámico (Sabour et al., 2017).

Los resultados que se han obtenido por la arquitectura basada en redes de cápsulas demuestran que se está muy al principio del estudio y desarrollo de estas redes, y aunque todo apunta a que con el tiempo se obtendrán mejores resultados, en este estudio hemos detectado que sólo mejora a las redes convolucionales en conjuntos de datos muy sencillos. También se observa que es una arquitectura computacionalmente muy costosa. Es, de todas maneras, una arquitectura base muy a tener en cuenta para seguir estudiándola y desarrollándola, ya que con más recursos computacionales y arquitecturas más sofisticadas, estudiadas y ajustadas podría llegar a dar muy buenos resultados.

2. Contexto y estado del arte.

En este capítulo se va a dar una visión histórica y teórica tanto de las arquitecturas basadas en redes convolucionales como las basadas en redes de cápsulas ya que en ambas está la base para hacer la comparativa. Se hará una pequeña explicación teórica, para luego hacer un recorrido por el tiempo para ver el desarrollo y evolución que han tenido las redes convolucionales, los problemas e inconvenientes que al lado de los logros han ido apareciendo, y cómo estos primeros han favorecido la búsqueda de alternativas a estas redes convolucionales, dando lugar a, por ejemplo, las redes de cápsulas. Con éstas también se da una pequeña introducción teórica, y luego se muestran los artículos donde se explica y justifica esta teoría más a fondo, junto con algunos de los trabajos más destacados hasta el momento.

2.1 Redes neuronales convolucionales

Las redes neuronales convolucionales también llamadas CNN (Basaveswara, 2019) son algoritmos que pertenecen al campo del *Machine Learning.* Estos algoritmos toman imágenes como entrada, detectan una serie de características dentro de cada imagen y en base a estas características son capaces de distinguir unas imágenes de otras, siendo capaces, por ejemplo, de realizar una clasificación.

La tarea de clasificación de imágenes en clases, por ejemplo, coches, camiones o motos, es la capacidad de ver una imagen y saber a qué grupo de los anteriores corresponde. Para los humanos es una tarea relativamente sencilla que se empieza a aprender siendo muy pequeños y de adultos es una tarea que prácticamente se hace de manera automática. Estas habilidades en clasificación, es decir, de reconocer patrones y generalizar en base al conocimiento que se tiene para poder decidir qué objeto se está viendo (a qué clase pertenece) y que para los humanos resultan tan sencillas, para las máquinas son tareas realmente complicadas.

```
(162, 239, 116) (158, 129, 148) (243, 135, 141) ..... (225, 202, 106) (252, 248, 124) (238, 255, 144)
(242, 187, 146) (186, 229, 120) (135, 250, 101) ..... (130, 158, 143) (183, 147, 146) (135, 199, 148)
     .....                                 .....                                 .....
     .....                                 .....                                 .....
(240, 129, 135) (187, 167, 114) (184, 138, 118) ..... (133, 149, 135) (153, 208, 140) (203, 225, 147)
(213, 127, 127) (176, 178, 118) (255, 155, 137) ..... (136, 133, 115) (231, 149, 142) (172, 162, 135)
```

FIGURA 2 LO QUE VE UNA MÁQUINA Y UN HUMANO.
Fuente: Elaboración propia

Los ordenadores no ven imágenes, las leen. Como se puede ver en la Figura 2 Lo que ve una máquina y un humano., en la imagen inferior aparece el personaje de los Simpson llamado Bart. Los humanos lo pueden reconocer por su característico color amarillo de dichos personajes, por su pelo, por una serie de características físicas que se ven. Sin embargo, en la imagen , en la parte superior, se puede ver lo que *ve* una máquina también: una serie de números que indican la intensidad de un pixel. En este caso, la máquina está leyendo una matriz de tamaño 64x64x3. Los dos primeros números indican el número de columnas y filas que hay, y el tercero, al ser una imagen a color, por cada pixel está leyendo tres valores (los valores de RGB). Si hubiera sido una imagen en blanco y negro, en vez de tres valores se tendría uno (la matriz sería de 64x64x1) indicando el último número la intensidad de gris correspondiente. Estos números que para nosotros de manera aislada no tienen sentido, para la clasificación de imágenes es todo lo que le llega a la máquina, es decir, al algoritmo. Y a partir de estos números nuestro algoritmo tiene que decidir con qué probabilidad la imagen pertenece a una clase u otra. Es decir, se pretende que el algoritmo diferencie entre todas las imágenes y sepa qué características hacen única a una imagen para saber si corresponde a la categoría de coches, la de motos o la de camiones. Cuando una persona ve un coche, sabe que es un coche, por ejemplo, porque tiene la característica que le define que son cuatro ruedas. De la misma manera, el algoritmo tiene que buscar los patrones básicos que le permitan reconocer de qué objeto se trata. Detectará primero bordes, curvas y líneas para luego ir construyendo objetos más complejos como círculos y a través de las capas superiores

construir objetos más abstractos, como ruedas. Esto es básicamente el funcionamiento de una red neuronal convolucional.

Las redes neuronales convolucionales se inspiran, como su nombre indica, en el funcionamiento biológico del córtex visual. La corteza visual tiene pequeñas regiones de células que son sensibles a ciertas zonas del campo visual. En un experimento que realizaron Hubel y Wiesel (Hubel & Wiesel, 1962), descubrieron que las neuronas están organizadas en grupos especializados, de manera que cada grupo de neuronas se activaba de manera distinta dependiendo de la información recibida. Estos grupos de neuronas están dentro de una arquitectura de manera que todos juntos pueden producir la percepción visual. Este sistema con tareas específicas, y componentes especializados como las neuronas de la corteza visual que buscan características, es el mismo sistema y es la base que usan los algoritmos de redes convolucionales.

Las capas convolucionales usan filtros, o kernels para buscar y detectar características. Son matrices cuadradas, por ejemplo, de 3x3 y esta operación de convolución consiste en el producto escalar del filtro por un tamaño igual en la imagen, y su suma es el valor de la nueva capa convolucionada. El filtro recorre toda la imagen de izquierda a derecha y de arriba abajo calculando todos los valores de salida de la capa convolucionada. La Figura 3 esquematiza la operación de convolución .

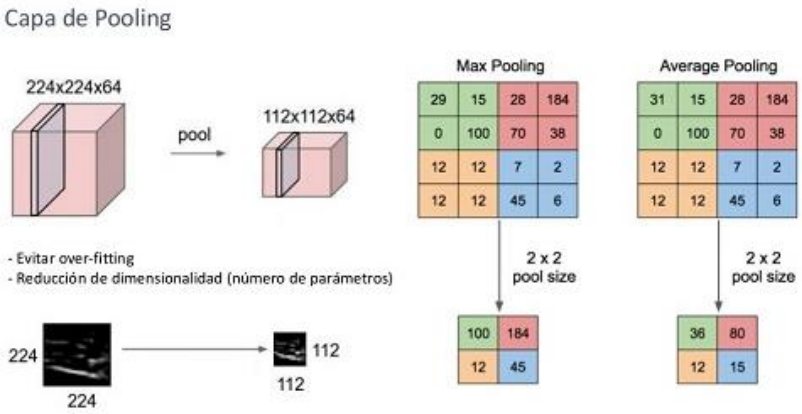

FIGURA 3. OPERACIÓN DE CONVOLUCIÓN.

Fuente: (Calvo, Diego, 2017)

Dicho de manera más abstracta, una operación de convolución consiste en tomar grupos de píxeles cercanos de la imagen de entrada, e ir operando matemáticamente (producto escalar) contra una pequeña matriz llamada kernel. Este kernel recorre todas las neuronas de entrada (de izquierda a derecha y de arriba abajo) generando una nueva matriz de salida.

Este proceso de aplicar el filtro, se suele realizar varias veces, dependiendo de la arquitectura. Por ejemplo, en una primera convolución se podrían tener 16 filtros, de manera que se obtendrían 16 matrices de salida. Estas matrices resultantes son los mapas de características (*feature map*). Se tendrían 16 imágenes nuevas filtradas dibujando cada una, una característica de la imagen original. Para saber el tamaño de salida de una capa, después de haber aplicado una capa de convolución se tiene la siguiente fórmula:

$$O = \frac{W - K + 2P}{S} + 1$$

Donde *W* es el número de píxeles (ancho) de la capa de entrada, *K* es el tamaño del filtro o *kernel* a aplicar, *P* es el padding (número de píxeles que se agregan alrededor de una imagen), y *S es el stride* (cuántas posiciones, filas o columnas de moverá el kernel).

Una vez aplicada una capa de convolución, con los filtros necesarios, se aplica la función de no linealidad *ReLU*. Esta función, llamada de activación, anula los valores negativos y deja los positivos sin variar:

$$f(x) = \max(x, 0) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \geq 0 \end{cases}$$

Esta función es la más usada en redes convolucionales, ya que tiene un buen desempeño con las imágenes.

Después, para reducir la dimensionalidad de las capas resultantes (*feature maps)*, y que no resulte inmanejable a través de la aplicación de los distintos *kernels*, se aplica una capa de *pooling,* la cual retendrá la información más relevante. Requiere la definición de un *spatial neighborhood* (indica el tamaño de la ventanta a las que se aplicará la reducción) y la operación a aplicar a los elementos de esa ventana, de la cual resultará un único elemento que será representativo de todos. En el caso de capas de *max pooling*, se elige el valor más alto de estas ventanas definidas. La técnica de *max pooling*, consiste en dividir la imagen en sub-matrices (por ejemplo, 2x2), y coger el máximo valor de cada sub-matriz. De esta manera, la imagen quedaría reducida a la mitad, como se puede ver en la Figura 4. Al aplicar esta capa de *max pooling*, sólo las neuronas más activas (máximos valores) se están pasando a la siguiente capa, con la consiguiente pérdida de información.

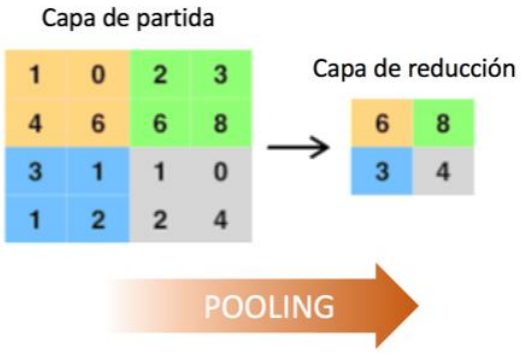

FIGURA 4. CAPA DE POOLING

Fuente: (Calvo, Diego, 2017)

Y aquí vendría uno de los grandes problemas de las redes convolucionales, y es que no son robustas frente a variaciones en los datos de entrada, como pueden ser modificaciones del fondo, rotaciones, cambios en la escala o diferentes puntos de vista. Mediante la operación de *max pooling* se puede conseguir una cierta invarianza frente a la traslación, ya que va reduciendo la capa, o *feature map*, porque se queda la información más relevante. Como normalmente se suelen emplear tamaños de ventana de 2x2, sólo se consigue una cierta invarianza a la traslación en capas muy profundas o próximas a la salida. Es decir, que no proporciona un aprendizaje robusto frente a distintas escalas o rotaciones. Una forma de conseguir que el modelo convolucional resulte más generalizable consiste en aumentar el *dataset* aplicando a sus imágenes una serie de modificaciones, de manera que estos nuevos ejemplos amplíen el cojunto de ejemplos con los que se entrenará el algoritmo.

Una vez aplicadas las operaciones de convolución y de *max pooling* correspondientes en la arquitectura, finalmente se llega a la última capa que es la capa clasificadora llamada *Fully connected layer (FC)*. A esta capa llegan la salida de las capas convolucionales que representan características de alto nivel de la imagen (por ejemplo, cuatro ruedas). La capa completamente conectada es una forma sencilla de aprender combinaciones no lineales de estas características. De hecho, aprende una función no lineal que toma un volumen de entrada (generado por las capas de convolución y *ReLU*) y lo convierte en un vector N dimensional (siendo N el número de clases a clasificar). Y cada número en este vector representa la probabilidad de una determinada clase. La capa conectada lo que hace es observar el resultado de la capa anterior (son los mapas de características de alto nivel) y determina qué características se relacionan más con cada clase en particular. Si el algoritmo predice que será un coche, tendrá valores altos en los mapas de características que representen cuatro ruedas.

Las redes neuronales convolucionales se entrenan con el algoritmo de *backpropagation* (Rumelhart et al., 1986). Estas redes, a partir de una entrada generan una salida. Al inicio de la red, cada neurona es multiplicada por un valor, llamado peso, que se calcula de manera aleatoria. Las neuronas, viajan por la red, realizando las operaciones correspondientes (convoluciones y *ReLU* entre otras). En la salida se calcula el error que se ha cometido, y se hace una vuelta hacia atrás, pasando de nuevo por todas las neuronas y calculando el error que ha cometido cada una de ellas, de manera que una vez que se vuelve al inicio, se puedan calcular y ajustar de nuevo los pesos de cada neurona, para realizar otra vez el entrenamiento (pasar por todas las capas y funciones definidas), e ir, iteración tras iteración de este algoritmo ajustando los pesos para que el error de la salida vaya disminuyendo.

Se puede resumir que las redes convolucionales son combinaciones de capas de convolución y de capas de *pooling* que extraen las características principales de cada capa. Y como última capa está la capa *fully connected*, o capa clasificadora que es la encargada de realizar las predicciones. La capa de *pooling*, además de reducir el coste computacional, controla el sobreajuste (*overfitting*). Esta reducción en la dimensión se traduce en pérdida de información y cuando estas redes se enfrentan a datos complejos, donde las jerarquías, el orden y la posición de las características son muy relevantes, cometen fallos, y llegan a ocasionar falsos positivos y falsos negativos debido a esa falta de información.

También hay que mencionar que cuando las redes convolucionales son muy profundas y complejas dejan de aprender apareciendo 3 principales problemas: la degradación/explosión del gradiente (*vanishing/exploding gradient*) y la maldición de la dimensionalidad (*curse of dimensionality*). El gradiente es el error que calcula el algoritmo de *backpropagation* (Rumelhart et al., 1986) durante el entrenamiento de la red, y se utiliza para actualizar los pesos en la red e ir mejorando las predicciones. Si este error se va acumulando, puede dar lugar a valores muy grandes haciendo que la red se vuelva inestable y deje de aprender. Se produce entonces la explosión del gradiente, o en inglés *exploding gradient*. Por el contrario, si este valor se va haciendo cada vez más pequeño, llega un momento en que, si se acerca mucho a 0, la red deja de aprender y se produce lo que se llama *vanishing gradient* (el gradiente desaparece).

La maldición de la dimensionalidad (*curse of dimensionality*) se produce cuando hay demasiadas características para tener en cuenta en una red neuronal, esto desemboca en un tiempo de computación muy alto, y en entradas que al final son irrelevantes o redundantes, y que hacen que la arquitectura no aprenda.

En 2011 Hinton presentó por primera vez el concepto de *Capsule networks* en el artículo *Transforming auto-encoders*, (G. E. Hinton et al., 2011), donde explica toda la teoría de estas

redes de cápsulas, pero no fue hasta finales de 2017, cuando por fin presentó un método para entrenarlas, con el que consigue mejorar algunos resultados obtenidos por las redes convolucionales. Sobre la manera de entrenar las redes de cápsulas se han presentado dos artículos *Matrix capsules with EM* (G. E. Hinton et al., 2018) y en *Dynamic route between capsules* (Sabour et al., 2017).

Hinton estudia cómo los gráficos 3D son capaces de representar una imagen por ordenador. La información de la imagen, tiene que ser leída y representada por éste. Esta información está guardada en una representación jerárquica de datos geométricos que tiene en cuenta las posiciones relativas de los objetos y está almacenada en memoria como conjunto de objetos y matrices que representan las posiciones relativas y la orientación de los objetos contenidos en la imagen. El programa correspondiente tiene que leer e interpretar esa representación y convertirla en una imagen en la pantalla. El cerebro hace justo lo contrario. De la información recibida por los ojos, deconstruye una representación jerárquica del mundo que nos rodea y trata de relacionarla con patrones y relaciones que ya ha aprendido y están guardados en el cerebro de manera que esta representación no depende del ángulo de visión que se tenga.

En la teoría de las redes de cápsulas para realizar el reconocimiento de objetos se guardan las relaciones jerárquicas de las posiciones de los objetos de la imagen y lo representan de manera numérica en un vector de pose (posición). Cuando estas relaciones se construyen en una representación interna de los datos, es cuando se facilita al modelo la labor de aprender y reconocer lo que hay en la imagen.

Es decir, que las características que están relacionadas entre sí debido a su orientación y situación espacial se agrupan en estas cápsulas. Estas cápsulas realizan complicadas operaciones internas en sus entradas, y luego encapsulan el resultado en un vector que contiene toda la información de las características para la salida de la capa. Sin embargo, en las redes convolucionales cada neurona tiene como salida un único escalar, resultado de aplicar los filtros a las distintas capas. Luego, en la capa de *max-pooling* se va seleccionando el número más alto de cada región, de manera que se conseguiría la invariabilidad de la capa. Esta invariabilidad implica, que un pequeño cambio en la entrada, no tiene que suponer un gran cambio en la salida. Es decir, si se cambia un poco el aspecto o posición de un objeto, las operaciones realizadas sobre ella por la red, no cambiarán esa agrupación máxima y la red seguiría pudiendo clasificar correctamente el objeto. Y aquí es donde está uno de los grandes problemas de estas redes. Estas operaciones, en la práctica, no terminan de funcionar de manera adecuada, porque al aplicar la capa de *max pooling* se está perdiendo información que puede ser muy importante, ya que no codifica, ni guarda las relaciones espaciales entre las características. Y un movimiento de rotación, por ejemplo, podría dar

lugar a un falso negativo. Pero las cápsulas de red, al guardar la información relativa a la posición de los objetos en un vector pueden reconocer el objeto al tener toda esa información guardada.

Estas cápsulas codifican en un vector la probabilidad de detectar una característica en la longitud del vector. El estado de la característica (dónde y cómo está colocada) es codificado como la dirección del vector. De manera que cuando una de las características cambia su posición, la probabilidad y el estado seguirán siendo los mismos, mientras que la dirección del vector cambiará.

Si se imagina un coche en 3D que se mueve por la imagen, la dirección del vector irá cambiando (el vector rotará), detectando el cambio en el coche, pero la probabilidad del vector y su longitud seguirán siendo fijos por lo que la cápsula seguirá detectando un coche. A esto lo llama Hinton una actividad con equivalencia: cuando un objeto cambia su estado a cualquier otra forma posible y la probabilidad de detectarlo no cambia (cambia la dirección del vector), y esto es lo que falta en las redes convolucionales.

2.1.1 Evolución de las CNN en la clasificación de imágenes

A continuación, se va a ver el desarrollo que han tenido estas redes a través del desafío anual *ImageNet Large Scale Visual Recognition* (ILSVRC) (J. Deng et al., 2009). En este desafío se presentan trabajos de todo el mundo, donde compiten por ver quién presenta el mejor y más preciso modelo para la clasificación de imágenes y detección de objetos sobre el conjunto de datos *ImageNet*(J. Deng et al., 2009). Este conjunto de datos es el mayor que existe categorizado, y de cada categoría hay una media de 1000 imágenes (llegando a haber más de 80000 categorías). Este desafío se presentó por primera vez en 2009, pero no fue hasta el 2012 cuando se empezaron a usar las CNN de manera más efectiva (hacían falta, entre otras cosas, grandes recursos computacionales). Se ha visto cómo desde ese momento, los resultados han sido cada vez más precisos disminuyendo la tasa de error de manera considerable. La Figura 5 ilustra en un gráfico esta disminución del error. Se puede apreciar cómo en el año 2012 al aparecer las redes convolucionales el error bajó más de un 45% con respecto a la edición anterior. Y una vez que se empezaron a estudiar y probar las redes convolucionales se ve cómo el error, año tras año va bajando de manera considerable, como en el año 2014 donde se consigue una reducción de más del 40% de error y en el 2015 casi del 50%.

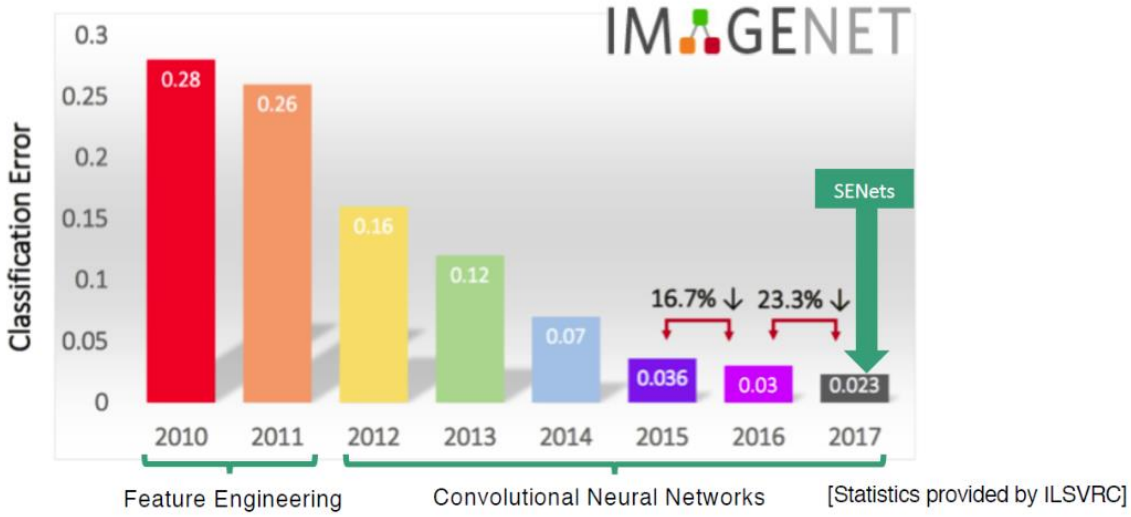

FIGURA 5. EVOLUCIÓN DE LA TASA DE ERROR EN EL DESAFÍO DE IMAGENET

Fuente: (Tsang, 2018b)

Se detallan a continuación los ganadores de este desafío cuyos trabajos han ido marcando el estado del arte en cada momento. Se empieza desde que apareció la primera red convolucional *Alexnet* en 2012 hasta la actualidad. También se estudiarán las novedades y mejoras que se han ido presentando en cada uno de los estudios que han sido ganadores de este desafío año tras año.

Alexnet

En el año 2012, con el modelo *AlexNet* (Krizhevsky et al., 2012), fue la primera vez que ganó el reto una arquitectura basada en CNN y la tasa de error experimentó una bajada considerable. En concreto *AlexNet* (Krizhevsky et al., 2012), consiguió que la tasa de error que estaba por encima del 25% pasara a ser menor del 17%. *AlexNet* (Krizhevsky et al., 2012) tiene una arquitectura mucho más compleja y profunda que las redes que se habían usado hasta el momento, como *LeNet* (LeCun et al., 1998b). Estaba compuesta por 5 capas convolucionales y 3 capas totalmente conectadas (FC *fully connected*), con 60 millones de parámetros y 650000 neuronas. El entrenamiento duró 6 días. Se realizó sobre dos GPU's, por eso en el gráfico de la Figura 6 se ve la arquitectura dibujada dividida en dos partes.

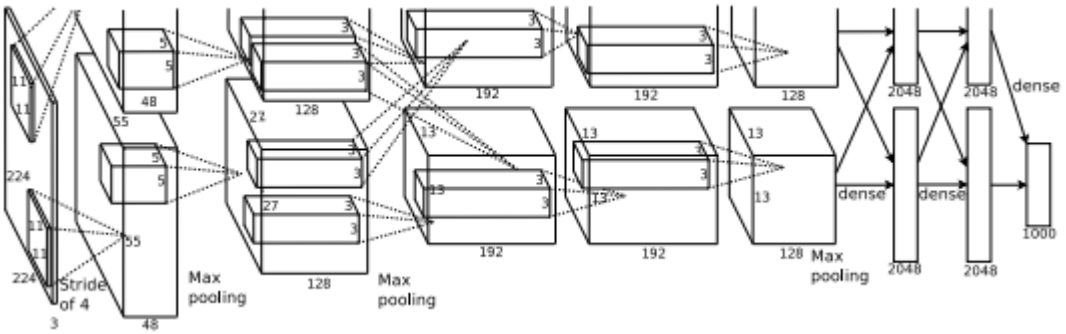

FIGURA 6. ARQUITECTURA ALEXNET

Fuente:(Krizhevsky et al., 2012)

Las capas de esta arquitectura están dispuestas de la siguiente forma:

CONV1 - POOL1 - CONV2 - POOL2 - CONV3 - CONV4 - CONV5 - POOL3 - FC6 - FC7 - FC8

Esta arquitectura, resultó mucho más profunda que sus predecesoras. La primera capa de convolución tiene 96 filtros (*kernels*) de tamaño 11x11x3. Como se ve en el esquema, las dos primeras capas, van seguidas de una capa de *max-pooling*. La tercera, cuarta y quinta capa convolucional, están directamente conectadas, mientras que a la quinta capa le sigue una de *max pooling*. Luego vienen las tres capas *fully connected*, donde la última capa aplica la función softmax (clasificadora) para realizar la predicción.

Después de cada capa de convolución y capa totalmente conectada se aplica la función no lineal ReLU. En la primera capa y la segunda capa convolución, la función *ReLU* precede a una función de normalización antes de aplicar la capa de *max pooling*. Las capas de *max pooling* se usan para reducir el tamaño de las capas, y reducir el tiempo de entrenamiento y mejorar el sobre ajuste. De hecho, consiguieron bajar el error top-1 en 0,4% y el error top-5 en 0,3%.

Otro descubrimiento importante en esta arquitectura, fue el uso de *ReLU* como función de activación en vez de *tanh* o *sigmoid*, que hasta el momento eran las que se usaban. Usando *ReLU* se aceleró el proceso de entrenamiento que fue hasta 6 veces más rápido sin perder precisión.

Para evitar el sobre ajuste usaron dos técnicas: una fue *data augmentation* (DA), que consiste en añadir perturbaciones a los datos originales para aumentar la cantidad de datos disponibles, como por ejemplo cambiarles la luz, ampliarlas, rotarlas o girarlas entre algunas. De esta forma se consigue así aumentar el banco de datos existentes. Y para regularizar, aplicaron la técnica de *dropout* que fue descubierta ese mismo año (Srivastava et al., 2014).

Esta técnica consiste en eliminar un pequeño porcentaje de neuronas (puede ser el 0,5%) de manera aleatoria, de forma que a la red le cueste más aprender (y no se aprenda los datos de entrenamiento).

Esta arquitectura marcó un hito en las tareas de visión artificial. A partir de este momento, en concreto para las tareas de clasificación de imágenes se fueron construyeron modelos de CNN con distintas arquitecturas, cada vez más complejos y más profundos y mejorando cada vez más los resultados. A pesar de ser una arquitectura algo antigua, se sigue usando, porque al no ser demasiado profunda, el tiempo de entrenamiento es relativamente corto.

ZFNet

En 2013, el ganador del reto ILSVRC fue ZFNet (Zeiler & Fergus, 2014). Esta arquitectura tiene una estructura muy similar a *AlexNet* (Krizhevsky et al., 2012), y mejoró la precisión estudiando la visualización de las capas de *AlexNet* (Krizhevsky et al., 2012).

Usaron lo que llamaron técnicas de deconvolución, es decir, revirtieron los procesos para poder visualizar las capas. Y se encontraron con dos problemas. Como se puede ver en la Figura 7:

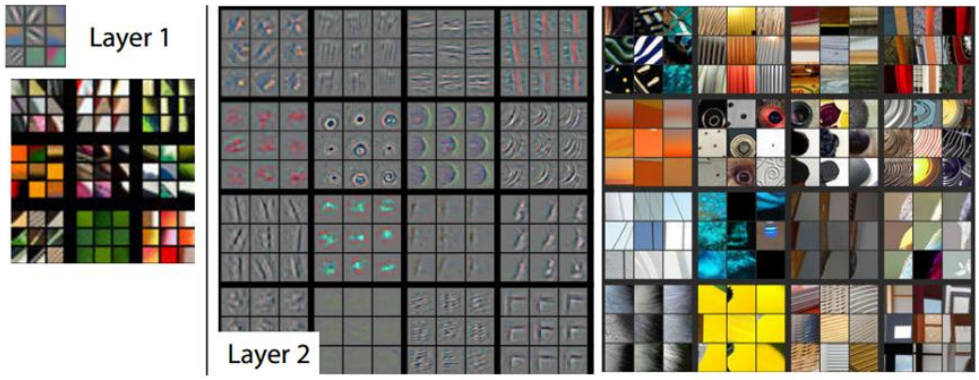

FIGURA 7. VISUALIZACIÓN DE LAS CAPAS DE ALEXNET

Fuente: (Deshpande, 2019)

Los filtros de la primera capa, tienen o bien frecuencias muy altas, o bien muy bajas (casi no tienen frecuencias medias), por lo que la red en estas capas aprende características que estén definidas en esas frecuencias (con la consiguiente pérdida de información). Y los filtros de la segunda capa, producen artefactos indeseados por la degradación de la imagen (*aliasing*) debido a las bajas frecuencias de la primera capa. Esto ha sido provocado por haber usado *stride* a 4 (demasiado grande). Las siguientes capas, van aprendiendo las características de manera más general a más detallada según las capas van ganando profundidad. Para

solventar estos dos problemas realizaron dos modificaciones: los filtros de la primera capa los cambiaron de tamaño, y pasaron de 11x11 a 7x7 y el tamaño del *stride* pasó de 4 a 2. *ZFNet* (Zeiler & Fergus, 2014) no sólo ganó el reto, sino que mostró de manera muy intuitiva cómo funcionan estas redes, proporcionando una nueva manera de entender estas redes y de poder mejorar el rendimiento.

VGGNet

Merece la pena mencionar al finalista del reto en 2014, que fue *VGGNet* (Simonyan & Zisserman, 2014). Esta arquitectura ha sido una arquitectura muy influyente, ya que reforzó el razonamiento intuitivo detrás de las CNN de que para que una de estas redes logre una representación jerárquica de los datos suficientemente buena, la red convolucional tiene que tener una estructura de capas profunda. Además, esta red también destacó por su simplicidad: está formada por 19 capas convolucionales (detrás de cada cual se aplica la función *ReLU*) con filtros de tamaño 3x3, con stride y pad a 1, y las capas de *max pooling* (2x2) con *stride* a 2. La configuración de los pesos está disponible de manera pública, y se ha utilizado en muchas otras aplicaciones y retos como base de extractor de características. Profunda y simple. Consiguió una tasa de error del 7.2%. En la Figura 8 se puede ver el dibujo de esta arquitectura.

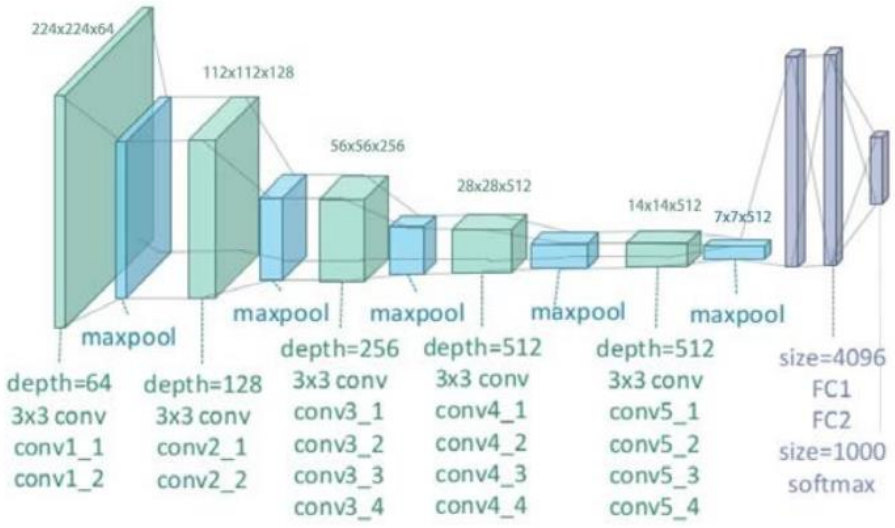

Illustration of the network architecture of VGG-19 model: conv means convolution, FC means fully connected

FIGURA 8. ARQUITECTURA DE VGGNET

Fuente:(Zheng et al., 2018)

GoogLeNet

La arquitectura de *GoogLeNet* (Szegedy et al., 2015) fue la ganadora del reto en 2014. Por el nombre, puede saberse que es de Google, y sus autores hacen un pequeño tributo al autor de la red *LeNet* (LeCun et al., 1998b), ya que parte de su trabajo se basó en esa red. A pesar de ello, esta red, tiene bastantes diferencias con sus predecesoras. Dejó el error en 6.7%. Pero si *VGG* (Simonyan & Zisserman, 2014) destacó por su simpleza, *GoogLeNet* (Szegedy et al., 2015) destaca, precisamente, por lo contrario. Esta red fue la primera red que se apartó del enfoque general que había hasta el momento, de simplemente apilar capas de redes convolucionales agrupadas una encima de la otra de manera secuencial. En la Figura 9 se puede ver la arquitectura que tiene este modelo. Se puede apreciar los distintos bloques y sub-bloques que conforman los distintos tipos de capas.

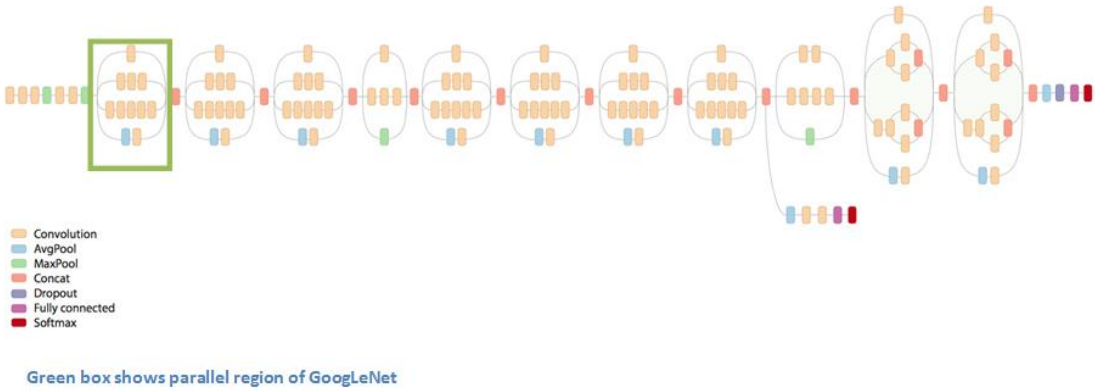

FIGURA 9. ARQUITECTURA VGG

Fuente: (Deshpande, 2019)

De un vistazo rápido a las y a la , la primera gran diferencia es la paralelización de procesos. Previamente, las arquitecturas eran secuenciales, pero aquí se introduce el concepto de módulo *Inception* (recuadro verde) sacado del trabajo *Network in Network* (Lin, Chen, & Yan, 2013). Un módulo *Inception* tiene la estructura descrita en la siguiente Figura 10:

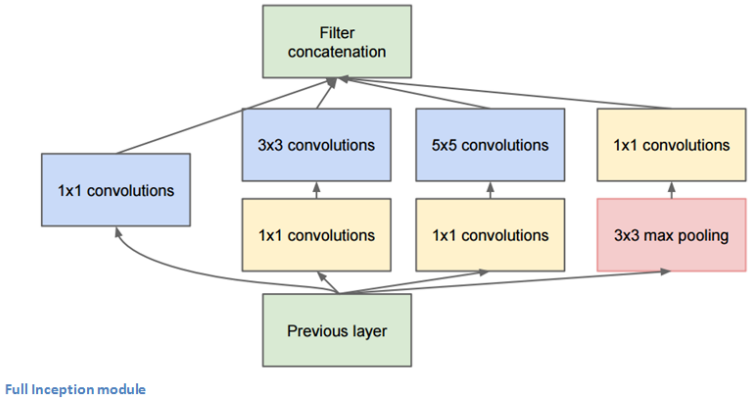

FIGURA 10. MÓDULO INCEPTION

Fuente: (Deshpande, 2019)

Tradicionalmente, después de cada capa convolucional, hay que decidir si se aplica una operación de pooling (para reducir el tamaño), o bien una de convolución, decidiendo también el tamaño de filtro de ambas. Pero los módulos *Inception*, permiten realizar todas estas operaciones en paralelo y con distintos tamaños de filtro. Así se puede ver que se aplican convoluciones con filtros de tamaño 1x1, 3x3 y 5x5, y cuando una imagen se procesa, se aplican diferentes convoluciones y diferentes *pooling*, ayudando a extraer diferentes tipos de características de la imagen. Los filtros de tamaño uno que se aplican antes de cada convolución de tamaño tres y cinco, son un método para reducir la dimensión del volumen, de manera que cuando se van a aplicar estos filtros, los tamaños de los volúmenes no resultan excesivos y así el número de parámetros se reduce drásticamente. Otra gran diferencia sobre sus predecesoras es que elimina la capa *fully connected* (FC). La capa FC, conecta todas las entradas con la capa final. Al eliminar esta capa, lo que hace es coger cada mapa de características de tamaño 7x7x1024, y lo convierte a uno de 1x1x1024, reduciendo el número de parámetros y consiguiendo una reducción del error en la precisión del 0,6%. Este método se llama *global average media*. Así, esta arquitectura está formada por 9 módulos *Inception* con 100 capas en total. Tiene doce veces el número de parámetros de *AlexNet* (12x60 millones) y el entrenamiento duró una semana sobre varias GPU's. Hay que subrayar que el elevado coste computacional y de memoria de esta arquitectura es debido al gran número de capas, filtros y parámetros que contiene la red. Y también, las grandes posibilidades de que la red caiga en *overfitting* debido a su complejidad.

Los autores consiguieron demostrar que con esta arquitectura tan creativa se puede conseguir una mejora en el rendimiento y en la eficiencia computacional en este tipo de redes tan complejas y profundas. De hecho, sentaron las bases para algunas arquitecturas realmente complejas que aparecerían en los años siguientes.

ResNet

En 2015 Microsoft presentó una CNN basada en bloques residuales, *ResNet* (He et al., 2016a). Fue ganadora del reto ya que bajó la tasa de error hasta el 3.57%. Consiguió con los bloques residuales y las 152 capas tener menos complejidad al entrenar que *VGGNet* (Simonyan & Zisserman, 2014). Con esta nueva arquitectura se batieron los records en clasificación, localización y detección de objetos. Además, marcó un hito, ya que los errores de las personas en tareas de visión artificial oscilan entre un 5% y un 10%, dependiendo de las habilidades y experiencias de cada persona. Y esta red tiene una precisión mayor. En la siguiente Figura 11 se puede ver su arquitectura:

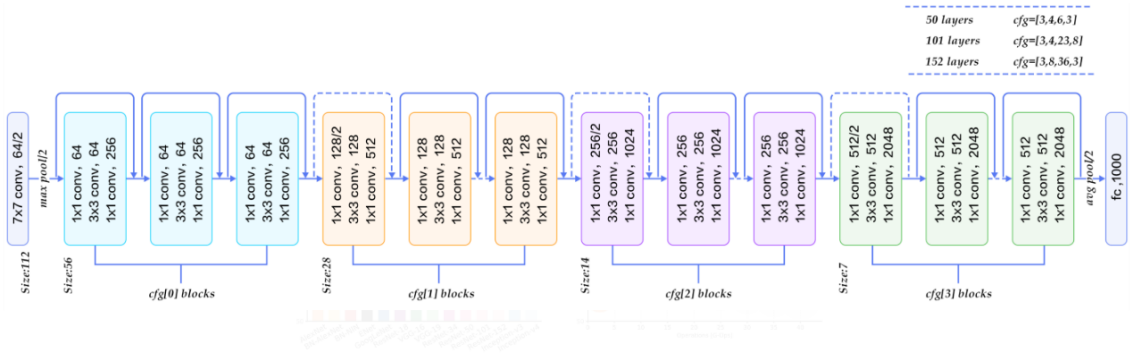

FIGURA 11. ARQUITECTURA RESNET

Fuente: (Basaveswara, S. K., 2019)

La idea subyacente en *ResNet*, es porqué apilar capas y capas, si a partir de cierta profundidad se ha demostrado, en contra de lo que pudiera parecer, que la red no siempre mejora. De hecho, aparecen problemas como el desvanecimiento del gradiente (*vanishing gradient*) y la maldición de la dimensionalidad (*curse of dimensionality*) y la red deja de aprender. También, con la profundidad de las capas la precisión llega un momento en el que se estanca y comienza a degradarse, con lo que se puede deducir que las redes convolucionales menos profundas aprenden mejor que sus "equivalentes" más profundas. Y aquí es donde nace la idea de los bloques residuales, como se ve en la Figura 12, ya que lo que hacen es saltar unas cuantas capas. Aumentan el número de capas introduciendo una conexión residual (con una capa identidad). Y los autores afirman que se mejora el proceso de aprendizaje, ya que esta capa pasa directamente a la siguiente.

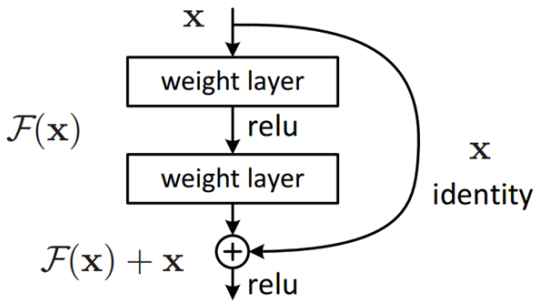

FIGURA 12. ESQUEMA DE UN BLOQUE RESIDUAL

Fuente:(Deshpande, 2019)

Se tiene el input x que en la red viaja a través de las capas conv-ReLU-conv-ReLU. Esta serie de operaciones dan como resultado F(x). Se llama a H(x) = F(x) + x. En una red convolucional tradicional se tiene que conseguir que F(x) y H(x) tengan el mismo valor. Así que en vez de intentar conseguir el valor F(x) directamente de x, con los bloques residuales se calcula el

valor que tiene que añadir F(x) a su entrada x. Es decir, el bloque residual lo que hace es añadir una pequeña alteración a la entrada x para conseguir una representación ligeramente alterada, mientras que, en las CNN tradicionales cuando se pasa de x a F(x), esta última es una representación que no tiene ninguna relación con su entrada original. Los autores afirman que es más sencillo optimizar el mapa residual, que el original que no está referenciado. Otro argumento que dan en favor de esta arquitectura es que durante la vuelta atrás con el algoritmo *backpropagation* (Rumelhart et al., 1986), el gradiente no se desvanece y se mantiene, gracias a las operaciones de suma que se van realizando a través de la red, las cuales van balanceando el gradiente.

Puntos interesantes para destacar sobre esta arquitectura, además del gran resultado conseguido en términos de precisión, son las 152 capas que la componen, y que después de las dos primeras capas el tamaño del volumen se ha reducido de 224x224 a 56x56. Los autores en su artículo afirman que un ligero aumento de capas en los bloques residuales da un resultado mejor, aunque probaron sobre una arquitectura de 1022 capas, que no dio el resultado esperado probablemente por un problema de *overfitting*. Y como último detalle, esta red (la original de 152 capas) estuvo entrenando sobre 8 GPU's durante algo menos de tres semanas.

Este tipo de arquitectura sobre bloques residuales han sido ampliamente estudiadas, ya que evitan la profundidad de la red, haciéndola más ancha, y convirtiéndola en una red más manejable, que sus correspondientes sin la arquitectura de los bloques residuales. Se van a ver unos cuantos ejemplos de ellos.

Arquitecturas ResNet

La arquitectura finalista del reto en 2016, *ResNeXt* (Xie et al., 2017) es una ampliación del modelo *ResNet* (He et al., 2016a), ya que añade una dimensión más, la cardinalidad a la arquitectura *ResNet* (He et al., 2016a), quedando los bloques de la siguiente manera:

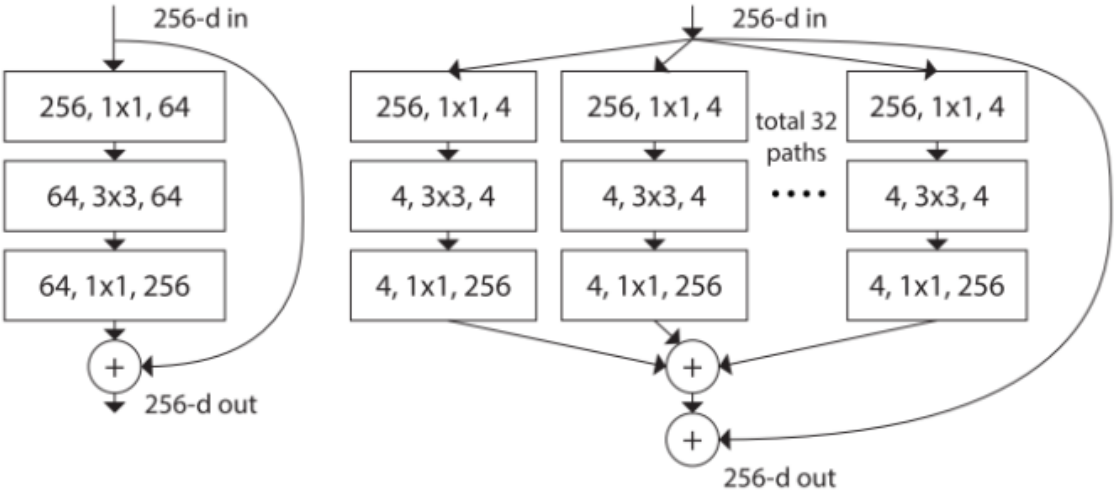

FIGURA 13. IZDA BLOQUE RESIDUAL RESNET Y DECHA BLOQUE RESNETX

Fuente: (Hitawala, 2018)

En la Figura 13 se puede ver a la izquierda un bloque *ResNet* (He et al., 2016a), y a la derecha su homólogo con una cardinalidad de 32. Consiguió dejar el error Top-5 en 3.03% (casi un 15% de mejora).

Este modelo fue muy costoso de entrenar a pesar de los diferentes intentos que se realizaron en el estudio *Evaluating ResNeXt Model Architecture for Image Classification* (Hitawala, 2018).

Trimps-Soushen

Volviendo de nuevo al reto, en el año 2016 la arquitectura ganadora la presentó *Trimps-Soushen* (Tsang, 2018a). *Trimps* es el instituto de investigación de tecnología para la seguridad china y *Soushen* traducido del chino significa “Dios de la búsqueda”. La arquitectura que presentó, y con la cual dejó el error por debajo del 3%, no presentaba ninguna tecnológica innovadora ni ninguna novedad. De hecho, no presentaron ningún artículo científico. Simplemente expusieron su trabajo en el *European Conference on Computer Vision* (University of Amsterdan, 2016) .

Usaron modelos pre-entrenados con *Inception-v3* (Szegedy et al., 2016), *Inception-v4*(Szegedy et al., 2017), Pre-activation *ResNet-200* (He et al., 2016b)y Wide *ResNet* (Zagoruyko & Komodakis, 2016) fusionando los resultados de los modelos. Observaron los resultados y se centraron en las diez categorías donde hubo más errores. Observando la Figura 14 se puede ver que no hay ningún modelo que supere a otro en ninguna categoría con respecto a los errores.

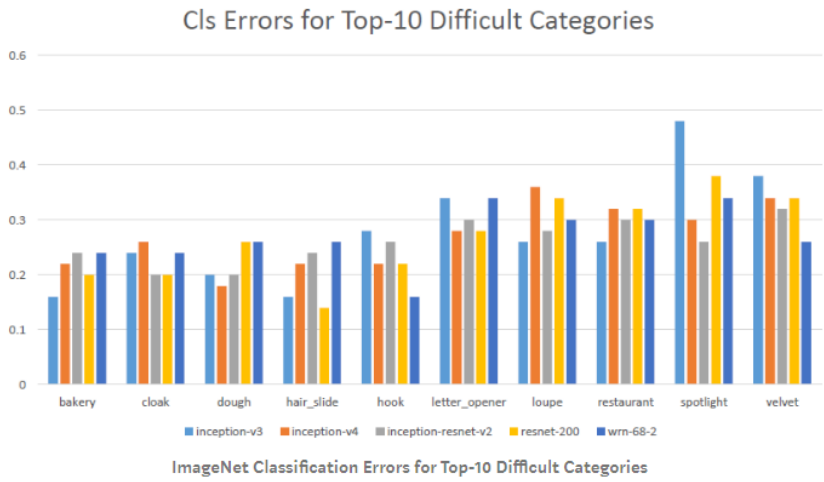

FIGURA 14. CLASIFICACIÓN DE ERRORES

Fuente: (Tsang, 2018b)

- El error de validación de los distintos modelos está entre 3,52% y 4,65%
- Entrenando los modelos de manera conjunta el error de validación queda en 2,92%
- Y el test error finalmente queda en 2,99%

Resulta llamativo el estudio de los errores que hicieron. Compararon el error top5, top10 y top20. Y vieron que entre estos dos últimos había un 1% de diferencia y miraron una por una las imágenes (había 1548) donde se había producido el error, llegando a la conclusión que la red sólo podía mejorar en dos categorías (*fine-grained* y *wrong*).

En la Figura 15 se pueden ver los errores que se produjeron en el modelo de *Trimps Shoussen* por categorías, contando el número de errores que se habían producido y su porcentaje correspondiente.

Error Categories	Numbers	Percentages(%)
Label May Wrong	221	15.16
Multiple Objects (>5)	118	8.09
Non-Obvious Main Object	355	24.35
Confusing Label	206	14.13
Fine-grained Label	258	17.70
Obvious Wrong	234	16.05
Partial Object	66	4.53

7 Error Categories

FIGURA 15. ERRORES DEL MODELO TRIMPS SHOUSSEN

Fuente:(Tsang, 2018b)

Se puede concluir en este apartado, que fusionando modelos se podría investigar y probar con otros muchos modelos, donde también se podrían mejorar estos resultados.

SE-Net

Un año más tarde, en 2017 el premio fue para la universidad de Oxford que presentó la arquitectura *SE-Net* (Hu et al., 2018). El error quedó reducido al 2,52% lo que representa una mejora del 25% con respecto a su predecesora presentada el año anterior por *Trimps-Soushen* (Tsang, 2018a). *Se-Net* introduce una nueva construcción de bloques que ayuda a la interdependencia que hay entre las capas, con no demasiado coste computacional añadido. Estos bloques pueden ser tipo inception o bien residuales. La idea básica es añadir parámetros a estos bloques de manera que la red pueda ajustar los pesos de cada capa. En una CNN los pesos de la red son los mismos cuando se crea el mapa de salida de las caraterísticas. *Se-Net* lo que hace es añadir un mecanismo para calcular un añadido a cada capa para que ésta mejore. Por ejemplo, podría añadir un parámetro más a la capa. Es decir, primero comprimen cada mapa de características en un valor, obtienendo un vector de tamaño n (número de capas convolucionales). Después se alimenta a través de una red neuronal de dos capas, que vuelve a generar un vector del mismo tamaño. Y esos n valores del mapa, son los que se pueden usar como pesos en los mapas de características, escalando cada canal en función de su importancia. En la Figura 16 se presentan dos posibles versiones de la arquitectura *Se-Net*, una con módulos *Inception* y otra con módulos *ResNet*.

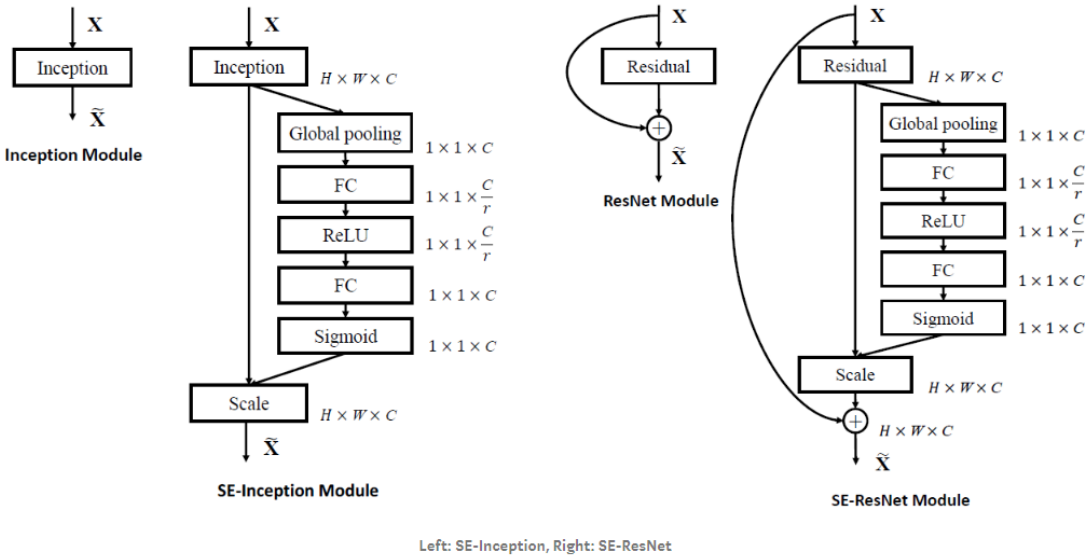

FIGURA 16. BLOQUES DE LA ARQUITECTURA SE-NET

Fuente: (Tsang, 2019)

Trabajos destacados

A continuación, se revisan una serie de estudios para hacer constar que existen multitud de trabajos que a pesar de no ganar el reto pudieran merecer una pequeña mención para ver la importancia que han ido teniendo estas arquitecturas las unas sobre las otras. Una de las formas de trabajo que se usan en infinidad de estudios es tomar como base arquitecturas con reconocido éxito, y modificarlas, o bien usar varias y luego unificar resultados, buscando siempre mejorar los resultados existentes.

En el mismo año en que apareció *ResNet* (He et al., 2016a), en 2016, apareció también otra arquitectura basada en bloques residuales, el estudio *Wider or deeper: Revisiting the resnet model for visual recognition* (Wu et al., 2019). Esta arquitectura estudia el comportamiento de *ResNet* (He et al., 2016a) y a través de una vista más intuitiva ayuda a comprender mejor su comportamiento y encontrar bases para futuros trabajos. Mejoró el rendimiento del modelo *ResNet-200* (He et al., 2016b) en el conjunto de datos de *ImageNet* y fue base de otros modelos, los cuales basándose en el nuevo enfoque consiguieron mejorar los resultados en conjuntos de datos como *Pascal*, *Voc*, *Pascal Context* y Paisajes Urbanos.

Otro ejemplo de arquitectura basada en *ResNet* (He et al., 2016a) es *Wide Residual Networks* (Zagoruyko & Komodakis, 2016). Este estudio mejoró los resultados existentes hasta el momento en los conjuntos de datos *COCO*, *SVNH*, *CIFAR* e *ImageNet*. Ese mismo año también salió el estudio *Residual networks of residual networks* (Zhang et al., 2017), que como su nombre indica añade un nivel más de bloque residuales logrando muy buenos resultados en *CIFAR-10*, *CIFAR-100* y en *SVSN*.

En 2017, el equipo de Google presentó *Neural Arquitecture Search* (Zoph & Le, 2016) en un esfuerzo por mejorar el estado del arte una arquitectura llamada *NAS*. En términos muy simples es un algoritmo que coge distintos tipos de bloques, como los que se muestran en la Figura 17, y una vez ha seleccionado los bloques, los combina, entrena y prueba la red. En función de los resultados se hacen ajustes en los bloques y en las arquitecturas diseñadas para ir mejorando los resultados.

- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv

FIGURA 17. BLOQUES USADOS EN ARQUITECTURAS NAS

Fuente: Elaboración propia.

Este algoritmo usa un controlador recurrente (*Recurrent Neural Network)* y muestrea los tipos de bloques agrupándolos y creando algún tipo de arquitectura. Estas arquitecturas suelen ser del tipo *ResNet* (He et al., 2016a) o *DenseNet* (Huang et al., 2017), pero con una configuración de bloques totalmente distinta. La arquitectura nueva se entrena y se valida, y con los resultados obtenidos, se vuelve alimenta al controlador recurrente para que genere mejores arquitecturas, seleccionando mejores bloques o haciendo mejores conexiones.

En el año 2018 presentaron una arquitectura en dos etapas: *HGR-Net, A fusión network for hand gesture recognition* (Dadashzadeh et al., 2019, p.). Los investigadores del presente estudio, proponen una nueva arquitectura de red neuronal convolucional (CNN) en dos fases, llamada *HGR-Net*, donde la primera etapa realiza una segmentación semántica precisa a nivel de píxeles para determinar las regiones de la mano, y la segunda etapa identifica el gesto de la mano.

La arquitectura de la etapa de segmentación se basa en la combinación de una red residual completamente convolucional y una agrupación de pirámides espaciales. Aunque la subred de segmentación está entrenada sin información de profundidad, es significativamente robusta frente a desafíos como las variaciones de iluminación y los fondos complejos.

La etapa de reconocimiento implementa una CNN de dos flujos que fusiona la información del RGB y las imágenes segmentadas al combinar sus representaciones profundas en una nueva capa completamente conectada antes de la clasificación. Los extensos experimentos en conjuntos de datos públicos de gestos con las manos muestran que la arquitectura profunda alcanza un rendimiento casi tan bueno como el de la última tecnología en segmentación y reconocimiento de gestos con las manos estáticas, en una fracción del tiempo de entrenamiento, tiempo de ejecución y tamaño del modelo.

Cabe destacar, que los investigadores emplearon una técnica aumento de datos que juega un papel importante en la obtención de una mayor precisión de reconocimiento. Señalan que el modelo mejor propuesto alcanza el estado del arte y rendimiento en el conjunto de datos *OUHANDS*. Además, evaluaron su modelo de segmentación en el conjunto de datos HGR1, donde superó a los métodos tradicionales de vanguardia

En 2017 se presentó un estudio *Wide Residual Networks* (Zagoruyko & Komodakis, 2016). Se demostraron que las redes residuales profundas pueden escalar hasta miles de capas y aún tienen un mejor rendimiento. Sin embargo, cada fracción de un porcentaje de precisión mejorada cuesta casi duplicar el número de capas, por lo que la capacitación de redes

residuales muy profundas tiene el problema de disminuir la reutilización de características, lo que hace que estas redes sean muy lentas para capacitarse.

Para abordar estos problemas, los investigadores llevaron a cabo un estudio experimental detallado sobre la arquitectura de los bloques *ResNet* (He et al., 2016a) , en base a lo cual proponen una arquitectura novedosa en la que disminuyeron la profundidad y aumentaron el ancho de las redes residuales. Denominaron a las estructuras de red resultantes amplias redes residuales (*WRN*) y mostraron que éstas son muy superiores a sus contrapartes delgadas y muy profundas de uso común.

De hecho, demostraron que incluso una simple red residual amplia de dieciséis capas de profundidad supera en precisión y eficiencia a todas las redes residuales profundas anteriores, incluidas las redes de profundidad de mil capas, logrando nuevos resultados de vanguardia en *CIFAR, SVHN, COCO*, y mejoras significativas en *ImageNet* (J. Deng et al., 2009).

En 2012 se presenta el trabajo (Krizhevsky et al., 2012) donde se compara el rendimiento de las redes neuronales profundas (*DNN*) con sujetos humanos en imágenes distorsionadas. Los investigadores demostraron que, aunque los *DNN* se desempeñan mejor o a la par con los humanos en imágenes de buena calidad, el rendimiento de *DNN* era aún mucho más bajo que el desempeño humano en imágenes distorsionadas.

Además, encontraron que hay poca correlación en los errores entre los *DNN* y los sujetos humanos. Esto podría ser una indicación de que la representación interna de las imágenes es diferente entre los *DNN* y el sistema visual humano. Estas comparaciones con el desempeño humano podrían usarse para guiar el desarrollo futuro de *DNN* más robustos.

Se puede apreciar, que las CNN cada vez son más complejas. Se buscan nuevas combinaciones, y nuevas operaciones sobre elementos como los bloques conocidos, o arquitecturas ya probadas, de manera que puedan surgir nuevos logros y mejoras en el estado del arte. Por eso, la nueva teoría de Redes de Cápsulas resulta tan interesante: es un nuevo enfoque de arquitectura y construcción de bloques (en este caso se asimilaría bloques a cápsulas) que enfatiza en no perder información desde un principio para poder aprovecharla y obtener mejores resultados.

Como se ha visto en la introducción, Hinton da un nuevo enfoque sobre estas redes y la manera en que aprenden los mapas de características. A continuación, se va a analizar los estudios presentados por su equipo, y otros que ya van saliendo a la luz presentando mejoras

sobre este primero, y sobre algunas muy conocidas CNN. Hay que tener en cuenta que este enfoque nuevo apenas lleva dos años en el mercado.

2.2 Redes de cápsulas

La idea de las cápsulas fue descrita por primera vez en 2011 por Geoffrey Hinton en el artículo *Transforming Auto Encoders* (G. E. Hinton et al., 2011). En este artículo se describe cómo las redes convolucionales son capaces de reconocer objetos, pero fallan a la hora de conocer su posición en el espacio. Esto es debido al uso de las capas de *max pooling*, que son las que proporcionan la invariabilidad traslacional y pierden esa información. En vez de centrarse en la invariabilidad, los modelos de redes neuronales deben centrarse en la equivalencia. Esto se puede hacer a través del uso de las redes de cápsulas, porque representan una sola entidad presente en la imagen y tienen como salida el vector de instanciación de los parámetros que representan las características de la entidad y la probabilidad de que la entidad exista dentro de un dominio acotado.

En una conferencia de Hinton en el MIT en 2014 (G. Hinton, 2014), describió cómo el cerebro humano es capaz de *renderizar* la información visual que recibe a través de los ojos, creando un esquema (tipo árbol) del objeto y sus partes. Así, mientras los programas de procesamiento de gráficos construyen la imagen empezando por la representación geométrica de las entidades, construyendo un modelo jerárquico a través de matrices invariantes, el cerebro humano hace justo lo contrario, creando los gráficos de manera inversa. Las representaciones de las entidades que el cerebro humano construye son independientes del punto de vista. Para construir un algoritmo de aprendizaje que encaje con las capacidades del cerebro humano, tiene que ser capaz de tener un punto de vista independiente, y los gráficos inversos tienen que ser el principal objetivo por conseguir.

Una cápsula es un pequeño grupo de neuronas que aprende a detectar un objeto en concreto en una región de una imagen. Como salida da un vector cuya longitud es la probabilidad de encontrar ese objeto en la imagen y su orientación da la pose del objeto. En la pose, se guarda toda la información relativa a posición exacta, tamaño, orientación y rotación entre otras. De manera que, si hay algún cambio en el objeto, como que esté redimensionado, rotado o girado, por ejemplo, la probabilidad del vector no cambiará, pero sí su orientación. Y de esta forma, se asegura mantener la equivalencia, es decir, ante un pequeño cambio de la imagen, el resultado no cambia, y sigue detectando qué objeto es.

Las redes de cápsulas, o CapsNet, se organizan en distintas capas. Las cápsulas primarias (las más bajas) reciben una pequeña parte de la imagen y tratan de encontrar la presencia y la pose de algún objeto simple, un rectángulo, un círculo o un cuadrado por citar algunos. Las

cápsulas en las capas más altas, llamadas cápsulas de enrutamiento detectan objetos más grandes y complejos, por ejemplo, una casa como la que hay en la Figura 18.

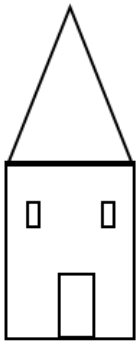

FIGURA 18. FIGURA CASA (RECTÁNGULO+TRIÁNGULO)

(Elaboración propia)

Las primeras capas de cápsulas están implementadas por capas convolucionales, que dan como salida un vector. Suponemos que la imagen contiene una cara y las primeras capas de cápsulas dan como salida los vectores u_1, u_2, y u_3. Se supone que las capas bajas detectan la boca, nariz y ojos, y la cápsula de salida, detecta una cara. Por ejemplo, u_1 sería el vector cuya longitud es la probabilidad de presencia de la boca y su dirección toda la información relativa a su situación espacial en la cara, u_2, el vector con la probabilidad y situación espacial de la nariz en la cara y u_3 contiene la de los ojos

Estos vectores, son multiplicados por su matriz de pesos correspondiente W que codifica las relaciones espaciales que hay entre las capas inferiores (ojos, nariz y boca) y la de salida (cara). Siguiendo con el ejemplo, W_{1j} sería la relación que hay entre la boca y la cara y guardaría que la boca está centrada en el cuarto inferior de la cara, que su tamaño es 8 veces menor que la cara y que la orientación es la misma para las dos, al estar en el mismo plano. El mismo tipo de información con respecto a los ojos y la nariz se guardaría en las matrices de pesos W_{2j}, W_{3j}. Una vez multiplicados los vectores, por sus pesos correspondientes, se obtendrían los vectores V_1, V_2 y V_3 donde V_1 representa dónde debe estar la cara con respecto a la posición de la boca, V_2 dónde debe estar con respecto a la posición de los ojos, y V_3 dónde debe estar la cara con respecto a la posición de nariz. Y si estos tres vectores apuntan en la misma posición y estado de la cara (la cápsula de salida), entonces deberá haber una cara.

Ahora faltaría por resolver, a qué cápsula superior envía la información las cápsulas inferiores. Y esto se hace por medio del algoritmo de enrutamiento dinámico.

Este algoritmo, decide cómo una cápsula i de una de las capas inferiores tiene que saber a qué capa j del nivel superior le manda la información del vector de salida. Y lo hará cambiando el valor del peso c_{ij} que representa el valor que multiplica al vector de salida de la cápsula i, y que será la entrada de la cápsula del nivel superior j. Y esto es lo que hace el enrutamiento dinámico por acuerdo. Las cápsulas inferiores enviarán la información al nivel superior de cápsulas a las que estén de acuerdo.

De esta forma, el enrutamiento dinámico lo que hace es decidir qué cápsula del nivel superior está de acuerdo con las entradas que le llegan de las capas inferiores. Y una cápsula del nivel inferior enviará el vector de salida a la cápsula de nivel superior que esté más de acuerdo con la entrada.

A continuación se muestra un ejemplo muy bien explicado (Geron, 2018). Hay que mirar la . Se supone, que las capas inferiores han detectado un cuadrado y un triángulo. Tanto uno como el otro pueden ser parte de una casa o de un barco. Dada la posición del rectángulo que está ligeramente girada hacia la derecha, la casa y el barco podrían estar también ligeramente rotados. Y con la posición del triángulo, la casa estaría del revés, mientras que el barco estaría ligeramente girado. Hay que recordar que tanto la presencia de las formas, como la relación que hay entre ellas, y entre cada una de ellas y la figura resultante, se aprende durante el entrenamiento. Si se observa, el triángulo y el rectángulo “están de acuerdo” con la pose del barco, mientras que están en desacuerdo con la de la casa. De nuevo, el proceso de manera más gráfica se puede ver en la Figura 19.

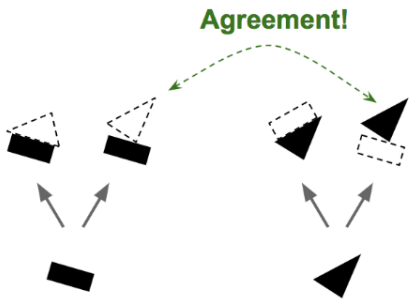

FIGURA 19. ALGORITMO ENRUTAMIENTO POR ACUERDO

Fuente: (Geron, 2018)

Una vez que se sabe que el triángulo y el cuadrado están de acuerdo en el barco, las salidas de las cápsulas del triángulo y el cuadrado se enviarán más a la cápsula del barco, y menos a la de la casa. Y así la cápsula del barco recibirá más información útil, y la cápsula de la casa

recibirá menos información que no le interese. Para cada conexión el algoritmo de enrutamiento dinámico, mantiene una ruta de pesos de manera que crecerá cuando haya acuerdos, y decrecerá cuando no los haya.

El algoritmo de enrutamiento por acuerdos, requiere de varias iteraciones de acuerdo-detección + actualizar enrutamiento, ya que ocurrirá con cada predicción no sólo con una, y no sólo en tiempo de entrenamiento. Este algoritmo funciona especialmente bien en escenas que están llenas de objetos. En el ejemplo de la se ve una imagen confusa, ya que se puede ver una casa del revés en medio, pero entonces se tendría un triángulo y un rectángulo que no encajarían con nada. Y aquí el algoritmo encontrará una mejor solución, ya que la mitad de abajo es un barco girado y la mitad de arriba una casa girada. Este algoritmo a través de distintas iteraciones por acuerdos, lo resolvería de esta manera. Se puede ver con más claridad en la Figura 20.

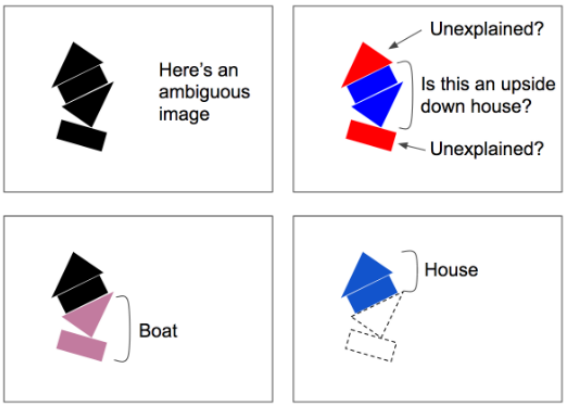

FIGURA 20. ALGORITMO ENRUTAMIENTO DINÁMICO

Fuente: (Geron, 2018)

Arquitectura de las redes de cápsulas

En esta sección se va a describir la manera en la que se construye una red de cápsulas. Esta arquitectura está basada en el artículo de (Sabour et al., 2017), y está construida para el *dataset* Mnist (LeCun & Cortes, 2010), formado por imágenes de 28x28 para clasificar diez dígitos, del cero al nueve.

En una arquitectura basada en redes de cápsulas, en un primer nivel se pueden distinguir dos partes: *encoder* y *decoder*.

La parte del *encoder* toma como entrada una imagen de m x m (28x28 en este caso) y aprende a codificarla en un vector n-dimensional. En este caso n=16 y es el número de parámetros de instanciación, es decir el número de características relativas a la imagen que va a aprender y guardar esta red. La salida de la red durante la predicción, será un vector de 10 dimensiones y es la salida de la capa DigitCaps. La parte del *decoder* está formado por tres capas: dos son capas convolucionales y la última es una capa *fully connected*.

Encoder:

- Capa1: Capa Convolucional
- Capa 2. Capa PrimaryCaps
- Capa3. Capa DigitCaps

Decoder:

- Capa 4. Capa *Fully connected*
- Capa 5. Capa *Fully connected*
- Capa 6. Capa *Fully connected*

Se va a ver ahora la arquitectura de las distintas partes de las redes de cápsulas.

Encoder

Capa 1. Capa convolucional.

Esta primera capa convolucional tiene como misión detectar las características básicas de la imagen de entrada. Está compuesta por 256 filtros (*kernels*) con un tamaño de 9x9x1, *stride* 1, seguida de una función de activación *ReLU*.

Capa 2. Capa PrimaryCaps.

Esta capa tiene 32 PrimaryCaps, cuyo trabajo es coger las características básicas que ha detectado la capa convolucional y generar combinaciones de estas características, es decir se compone de mapas de características de cápsulas, donde cada cápsula genera un vector de activación. La función de *squash* se aplica para calcular los vectores de salida finales de esta capa. Cada cápsula en esta capa, es decir, cada PrimaryCaps, es responsable de detectar una característica simple. Esta segunda capa, contiene las 32 PrimaryCaps que trabajan de manera muy similar a una capa convolucional. Cada cápsula aplica ocho filtros 9x9x256 con stride 2 al volumen de entrada 20x20x256, y como salida tiene un tensor de tamaño 6x6x8. Como hay 32 cápsulas, el volumen de salida será un vector de dimensión 6x6x8x32.

Capa 3. Capa DigitCaps.

Esta capa está formada por diez DigitCaps, una por cada dígito a detectar. Cada cápsula tiene como entrada el vector de tamaño 6x6x8x32. Se puede ver como un vector de 8 dimensiones

de tamaño 6x6x32, que serían 1152 vectores de entrada en total. Cada uno de estos vectores obtendría su matriz de pesos correspondiente de 8x16 que asigna el espacio de entrada de 8 dimensiones al espacio de salida de la cápsula de 16 dimensiones.

Esta es la capa más compleja y es responsable de detectar características y combinaciones de las características que vienen de la capa de las PrimaryCaps. De manera general, en esta capa habría una cápsula por cada salida de las clases del *dataset*.

En la Figura 21 se puede ver la arquitectura dibujada de un *encoder.*

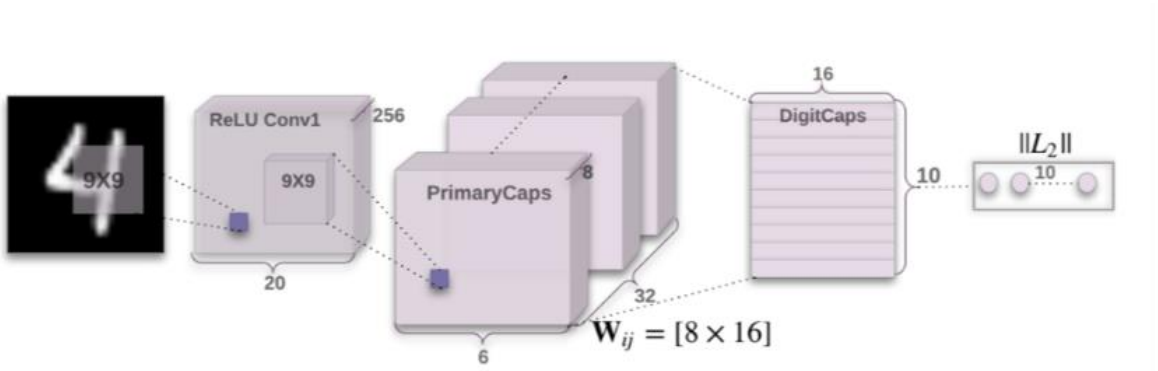

FIGURA 21. ARQUITECTURA DE UN ENCODER.

Fuente: (Sabour et al., 2017)

Decoder

El *decoder* coge el vector de 16 dimensiones correcto que proveniente de la DigitCaps y aprende a decodificarlo en una imagen de entrada, en este caso, en una imagen de un dígito. Hay que fijarse que coge el vector correcto, ya que durante el entrenamiento ha desechado los incorrectos. El *decoder* se usa como un regularizador, toma la salida correcta de la capa de DigitCaps y aprende a recrear una imagen de, en este caso, de 28x28 píxeles, con la función de pérdida con la distancia euclídea entre la imagen reconstruida y la imagen de entrada. El *decoder* obliga a las cápsulas a aprender características que sean útiles para poder reconstruir la imagen original, de manera que esta imagen quede bien construida. En la Figura 22, se pueden ver imágenes reconstruidas.

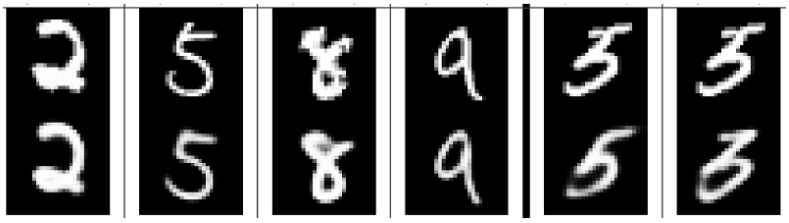

FIGURA 22. IMÁGENES RECONSTRUIDAS POR EL *DECODER.*

Fuente:(Sabour et al., 2017)

Las siguientes tres capas que forman el decodificador, son tres capas totalmente conectadas, con un funcionamiento similar. Cogen la salida del nivel inferior, se pondera y se dirige a cada neurona de la capa superior totalmente conectada como entrada. En la primera capa hay 16x10 entradas, que se dirigen a cada una de las 512 neuronas de esta capa.

Capa 4. Capa *fully connected*

Entrada: 16x10

Salida: 512

Capa 5. Capa *fully connected*

Entrada: 512

Salida: 1024

Capa 6. Capa *fully connected*

Entrada: 1024

Salida: 784, que es el tamaño de las imágenes decodificadas, 28x28.

Se puede ver un esquema en la Figura 23.

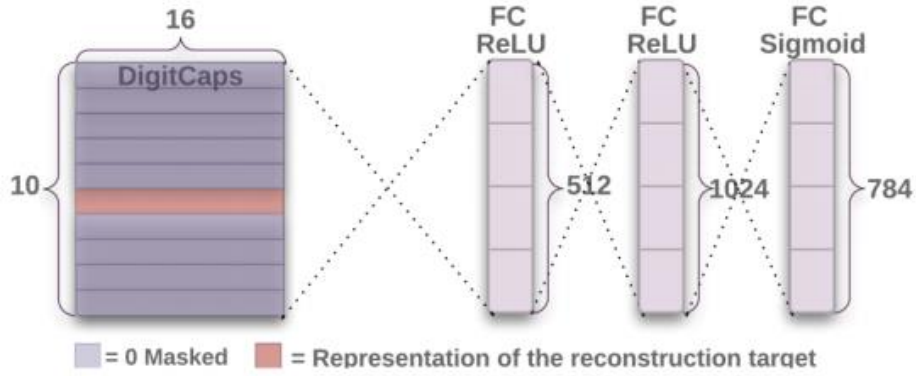

FIGURA 23. DECODIFICADOR DE UNA RED DE CÁPSULAS.

Teoría sobre redes de cápsulas.

Como ya se ha mencionado, en el artículo Transforming Auto-encoders (G. E. Hinton et al., 2011) se definen y explican por primera vez de las redes de cápsulas. Se presentó como una mejor opción a las técnicas que había (y que de hecho siguen estando vigentes) para la clasificación y/o detección de imágenes y objetos. Por un lado, existen los algoritmos basados en redes neuronales que usan varias capas de neuronas como detectores de características y reproducen las salidas en un único escalar. Por otro, están los algoritmos diseñados a mano, tipo SIFT (Lowe, 2004), cuya salida es un vector que incluye la representación explícita de la pose de cada característica. Y así presentan una nueva forma de diseñar redes neuronales para aprender características pero que generen como salida un vector que almacene toda la información relativa a la posición de las características, y argumentan que será una manera más fácil de manejar toda la información relativa a las variaciones de escala, iluminación, rotación y posición que los métodos que existen actualmente. También mejorará todas las funciones como SIFT, ya que este tipo de redes se adaptan a las características del dominio.

Este artículo se basa en las deficiencias que presentan las arquitecturas de redes neuronales convolucionales. Estas arquitecturas no ponen el foco en la invarianza de la actividad de las neuronas, y esto puede considerarse erróneo, ya que una de las consecuencias de guardar toda la información de la salida en un único escalar (capas de *max-pooling*), es que la información queda muy resumida y se pierde información muy importante de esas características. Sin embargo, si estas neuronas que tienen como salida un único escalar, se

agrupasen en grupos (cápsulas) locales, se podría, realizando los cálculos internos correspondientes con la información de entrada, lograr un vector de salida de manera que sea posible guardar toda la información espacial relativa a esas entradas. Cada cápsula aprenderá a reconocer una característica en una región limitada de la imagen, y generará una vector de salida donde se guardará la probabilidad de existencia de esa característica como un conjunto de datos (parámetros de instanciación) que entre otros pueden ser, la posición precisa, iluminación, rotación y/o escala. Si la cápsula funciona correctamente, entonces la probabilidad de que el objeto exista será invariante, y no afectará al resultado cualquier variación sobre él, como por ejemplo una rotación, un cambio de tamaño, un nuevo movimiento y/o un cambio en la iluminación. Lo que si varía, son los parámetros de instanciación de la cápsula, como el tamaño, la luz, el movimiento o la rotación, ya que en esos parámetros es donde se codifica toda esa información que va cambiando, pero que al final, no tiene que influir en si el objeto está o no está presente.

Una ventaja de trabajar con estos vectores de salida de las cápsulas, es que así aprenden a reconocer el todo (de la imagen) a partir de las partes (de la imagen). Si una cápsula aprende a reconocer la pose de su objeto (parte de la imagen), entonces será posible determinar si los objetos representados por dos cápsulas A y B tienen una relación espacial correcta para que se active una cápsula del nivel superior C. Si por ejemplo A representara una boca y B una nariz, cada una hará la predicción de su pose para la cara. Si las predicciones concuerdan, ambas estarán en la relación espacial correcta para formar una cara. Es decir, sus vectores apuntarán a la misma cara C del nivel superior de cápsulas. Trabajar de esta forma permite que el conocimiento de las relaciones entre la parte y el todo, el cual es invariante, esté representado por las matrices de pesos, mientras que los parámetros de instanciación de los objetos observados y sus partes equivalentes esté representado por las actividades neuronales.

Para poder trabajar de esta forma, y que las cápsulas tengan ese conocimiento entre la parte y el todo, las cápsulas tienen que aprender a sacar toda esta información, relativa a la pose, de la información contenida en los píxeles, su intensidad. Y al igual que en los humanos, que hay una traducción directa entre la información captada por la retina y la corteza visual, que tiene acceso a la información no visual sobre los movimientos oculares, las cápsulas aprenden de una manera sencilla pares de imágenes transformadas ya que la red neuronal tiene acceso directo a las transformaciones no visuales que se han producido.

Y la pregunta que se plantea es, ¿cómo se genera la primera capa de cápsulas? Y la respuesta es por medio del diseño de un “transforming auto-encoder”, un transformador auto-codificador.

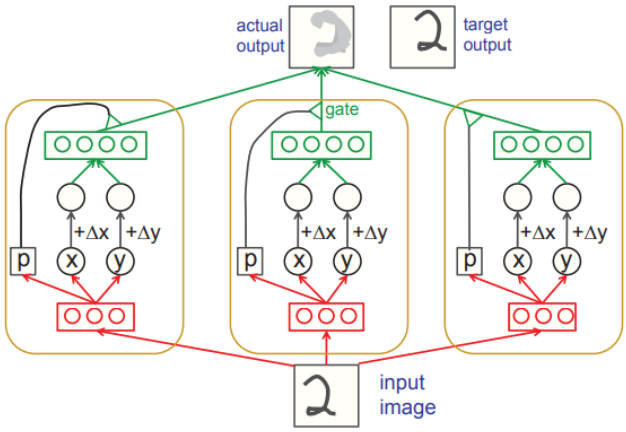

FIGURA 24. *TRANSFORMING AUTO-ENCODER*

Fuente: (G. E. Hinton et al., 2011)

La Figura 24 muestra tres cápsulas de un transformador auto-codificador que modela traslaciones. Cada cápsula tiene tres unidades de reconocimiento (rojo) y cuatro unidades de generación. Los pesos en las conexiones se aprenden con el algoritmo de backpropagation (Rumelhart et al., 1986).

La idea es que por cada entrada se genera una salida distorsionada, sumándole a cada par de puntos (x, y), píxeles, un incremento (Δx e Δy) y generando una probabilidad p. De manera que la salida es una imagen cambiada, con su probabilidad. Con más detalle, cada cápsula tiene sus unidades de reconocimiento. Estas unidades son una capa oculta que generará la terna x,y y la probabilidad de existencia p. Una vez hecho esto, cada cápsula tiene sus unidades de generación que cogerán la terna anterior y la transformarán en x + Δx y en y + Δy, sin cambiar el valor de la probabilidad p. Esta contribución que realizan las unidades de generación se multiplica por p, de manera que las cápsulas inactivas dejan de tener efecto.

En la Figura 25 se ve un ejemplo. Se ha utilizado un transformador auto-codificador con 25 cápsulas, donde cada una tiene 40 unidades de reconocimiento y 40 unidades de generación. En la primera fila están las imágenes de entrada. En la del medio, las imágenes de salida (después de pasar por las unidades reconocimiento y de generación) y en la tercera fila la imagen transformada correcta, después de pasar por el algoritmo de backpropagation (Rumelhart et al., 1986).

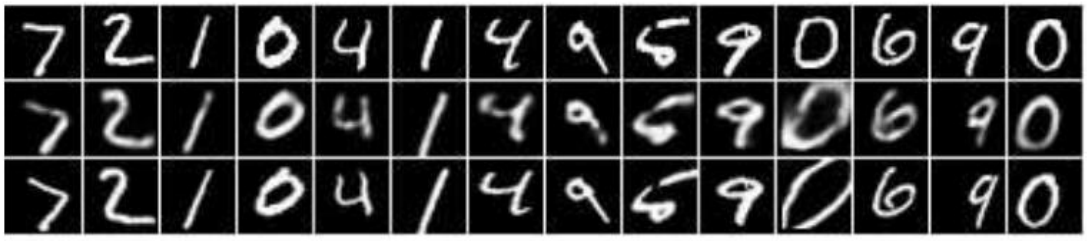

FIGURA 25. TRANSFORMADOR AUTOCODIFICADOR. IMÁGENES DE ENTRADA, SALIDA Y TRANSFORMADA

Fuente: (G. E. Hinton et al., 2011)

Si a cada cápsula se le dan 9 valores reales, en una matriz 3x3, se puede entrenar un transformador auto-codificador para predecir una transformación completa 2-D con traslación, rotación, escalado y corte. En este caso a la salida de las unidades de reconocimiento, se le aplicaría una matriz transformada en las unidades de generación para predecir la imagen de salida.

Estas redes al generar sus propias imágenes transformadas, necesitarán de muchos menos ejemplos para entrenar.

Una vez definidas las cápsulas, cómo funcionan y el objetivo a conseguir, llega el momento de entrenar estas arquitecturas. Pero no fue hasta el año 2017 cuando aparece el artículo *Dynamic Routing Between Capsules* (Sabour et al., 2017) donde se describe e implementa un método, el primero, para entrenar las redes de cápsulas llamado enrutamiento dinámico entre cápsulas, o también enrutamiento por acuerdos.

Los investigadores asumen que la información de la imagen tiene que estar guardada en una representación jerárquica de datos geométricos que tiene en cuenta las posiciones relativas de los objetos y que estará almacenada en memoria como conjunto de objetos y matrices que representan las posiciones relativas y la orientación de los objetos contenidos en la imagen, de manera que toda la información se conserve y no se pierda. Tiene su base en la visión humana. De esta forma, se diseña una estructura de árbol para guardar las imágenes con sus detalles, que se grabará en memoria como una estructura tipo red neuronal multicapa. Cada capa está dividida en pequeños grupos de neuronas llamadas cápsulas y cada nodo en el árbol se corresponde con una cápsula. Usando el algoritmo de enrutamiento iterativo, cada cápsula tiene que elegir qué cápsula del nivel superior será su padre en el árbol. Para las capas altas del sistema visual este procedimiento iterativo resolverá la pertenencia de las partes al conjunto entero. La actividad de las neuronas dentro de una cápsula representará varias propiedades de la parte que esté presente en la imagen. Estas propiedades incluyen

diferentes parámetros, tales como la pose (posición, tamaño y orientación), deformación, velocidad, albedo, tonalidad y textura entre otras. Y una propiedad muy especial es la existencia de esa parte en la imagen. La forma en que nos representan esta existencia es usando una entidad lógica cuya salida sea la probabilidad de que la parte existe. Esta entidad se representa como un vector cuya longitud es esta probabilidad. Se aseguran de que este valor esté entre 0 y 1 aplicando una función de no linealidad que no cambia la orientación del vector, pero sí que lo escala para que esta probabilidad se quede entre esos dos valores. De esta forma, el vector salida de la cápsula hace posible usar un potente algoritmo de enrutamiento para asegurar que la salida de la cápsula es mandada al "padre" apropiado en la capa superior de cápsulas. En un primer momento, el vector se envía a todos los posibles "padres", pero se reduce mediante los coeficientes de acoplamiento que suman 1. Para cada posible padre, la cápsula calcula un vector de predicción multiplicando el vector por una matriz de pesos. Si este vector resultante tiene una predicción suficientemente grande, existe una retroalimentación de arriba a abajo de manera que el coeficiente de acoplamiento de ese padre aumenta y para el resto lo disminuye. De esta forma se aumenta la contribución que esa cápsula hace a ese padre, aumentando más el producto escalar de la predicción de la cápsula con la salida del padre. Este método debería ser más efectivo, ya que en el enrutamiento por agrupación máxima (*max pooling*) se ignoran todos los detectores de características, menos los más activos, con la consecuente pérdida de información. Todas las capas de cápsulas menos la última son convolucionales. Como en las CNN el nivel más alto de las cápsulas aprende las regiones más grandes de la imagen, aunque a diferencia de como sí se hace en las capas de *max pooling*, en este caso no se deshecha información sobre la posición precisa de la entidad dentro de la región. Las cápsulas de las capas más bajas guardan la información del lugar de la cápsula correspondiente. Y según se asciende en la jerarquía más información de la imagen es guardada y codificada en el vector salida de la cápsula.

Una diferencia muy importante entre estas redes de cápsulas y las redes normales es que la activación de la cápsula está basada en la comparación entre múltiples entradas de la predicción de su pose, mientras que en una red neuronal normal está basada en la comparación de una única entrada del vector actividad y su vector de pesos aprendido.

Al poco tiempo de aparecer este artículo (menos de una semana), se publicó otro estudio también sobre enrutamiento dinámico, *Matrix capsules with EM routing* (G. E. Hinton et al., 2018) que también fue enviado al ICLR, International Conference on Learning Representations para su revisión. Es el artículo sobre teoría de cápsulas más reciente. Explica un nuevo algoritmo llamado *Expectation-Maximization*, EM, de máxima expectación, para entrenar una nueva versión de cápsulas. Las nuevas cápsulas están formadas por la

probabilidad de que la entidad exista y por una matriz 4x4 donde se aprende la relación que hay entre la entidad y el observador.

Cuando se hace el enrutamiento se asume que cada cápsula en la capa i, se activa porque es parte de algún “total” y pertenece a otra cápsula en la capa i+1. En este paso, se asume que hay variables latentes que explican de qué “total” proviene nuestra información y se trata de inferir la probabilidad de que cada matriz de salida provenga de la característica del nivel superior i+1. Esto se reduce a un problema de aprendizaje no supervisado, y se resuelve con el algoritmo de máxima expectación. Este algoritmo maximiza la probabilidad de que nuestros datos, la salida de la capa i, se expliquen en las cápsulas de la capa i+1. Este algoritmo es iterativo y consta de dos pasos, que se repiten tres veces. Los dos pasos de los que consta son: la expectativa y la maximización. El paso de expectación asigna probabilidades a cada cápsula en la capa i que es explicada por la cápsula en la capa i+1. Luego, el paso de maximización decide qué cápsulas en la capa i+1 están activas y si ven un acuerdo significativo de entradas, decide cuál es la entrada a la cápsula calculando un promedio de las entradas enrutadas.

Una vez visto qué son las cápsulas, y cómo se entrenan en sus distintos métodos, se va a ver una recopilación de trabajos y estudios.

Trabajos destacados basados en redes de cápsulas.

El primer trabajo que se va a revisar es *A capsule network for traffic speed predicion in complex road* (Kim et al., 2018). No trata específicamente sobre imágenes, sino sobre el conjunto de los datos de tráfico como tal. En este estudio, tratan los datos de tráfico espacio-tiempo como si fuera una imagen donde estos datos se expresan en un espacio 3D donde los ejes son el espacio y el tiempo, haciendo así el símil entre datos de tráfico e imágenes.

FIGURA 26. TRÁFICO DE SANTANDER.

Fuente: (Kim et al., 2018)

En la Figura 26, se puede ver representado el tráfico del centro de la ciudad de Santander. Los detectores de velocidad están marcados en rojo.

El artículo propone una solución basada en aprendizaje profundo para la predicción de la velocidad del tráfico en redes viales complejas. A pesar de que la cantidad de datos de los que disponen, no son muy elevados (los sensores de velocidad no están por toda la red donde se analizan los datos), proponen una red neuronal de cápsulas, evitando la operación de *max pooling*, y utilizando el algoritmo de enrutamiento dinámico. Los autores consiguen una mejora del 13% con respecto a una CNN, usando los datos reales de tráfico de la ciudad de Santander.

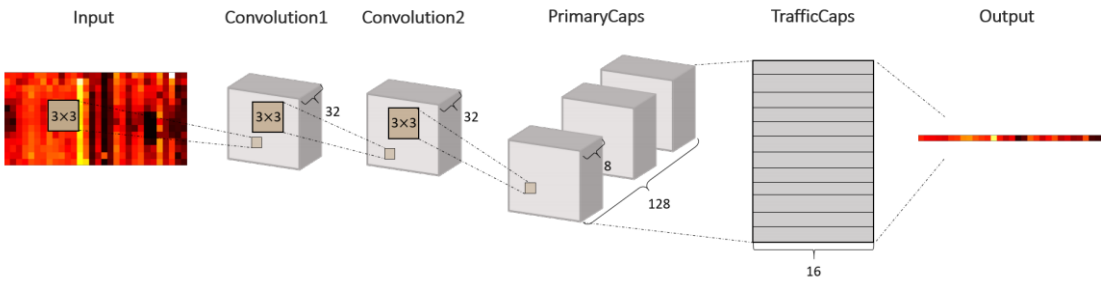

FIGURA 27. ARQUITECTURA PARA LA PREDICCIÓN DE LA VELOCIDAD DEL TRÁFICO.

Fuente: (Kim et al., 2018)

En la Figura 27 se puede ver la arquitectura propuesta. Consta de dos primeras capas de convolución que convierten la imagen espacio-tiempo en unidades para los detectores de características locales y que serán usadas como entrada a las cápsulas primarias de la red.

La tercera capa, llamada PrimaryCaps, es otra capa convolucional, compuesta por 128 capas a las que se les aplicará un kernel 3x3. Todas las operaciones de convolución tienen un *stride* de 1 y *zero padding*. Cada cápsula en la capa primaria da como salida un vector de 8 dimensiones, y comparten sus pesos entre sí. La capa final (a la que llaman TrafficCaps) tiene una cápsula con un vector de 16 dimensiones por cada segmento de carretera. El algoritmo de enrutamiento dinámico se aplica entre las PrimaryCaps y la TrafficCaps con 3 iteraciones. Captura la relación entre todas las cápsulas de la PrimaryCaps, donde cada cápsula representa la velocidad en un segmento de la carretera. De esta forma cada característica local puede contribuir a las cápsulas de la capa TrafficCaps. La longitud de cada vector de las TrafficCaps es la velocidad en el segmento de carretera correspondiente.

Layer	Parameter	Activation
Convolution1	(32, 3, 3)	ReLu
Convolution2	(32, 3, 3)	ReLu
PrimaryCaps	(128, 3, 3) Capsule size 8	ReLu -
TrafficCaps	Capsule size 16	-

FIGURA 28. CAPAS Y PARÁMETROS DE LA ARQUITECTURA PARA LA PREDICCIÓN DEL TRÁFICO

Fuente: (Kim et al., 2018)

La Figura 28 muestra la arquitectura y los parámetros usados.

El siguiente artículo que se presenta, *Hyperspectral image classification with capsule network using limited training samples* (F. Deng et al., 2018) presenta una arquitectura de dos capas de redes de cápsulas entrenada con un número de ejemplos limitado para clasificar imágenes hiperespectrales (HSI), inspirada en los modelos de aprendizaje profundos más superfeciciales. Han usado dos conjuntos de entrenamiento de imágenes HSI reales donde hay ejemplos de toda complejidad. El principal objetivo de este estudio es crear un paradigma comparable entre CNN y CapsNet, eliminando las diferencias y dejando ambas arquitecturas muy parecidas, con la diferencia de la arquitectura base. También han hecho un intento de mejorar CapsNet para conseguir los mismos resultados que han obtenido con la CNN. Y dado que las arquitecturas más profundas no siempre dan los mejores resultados, han diseñado un modelo más bien pequeño y poco profundo para ambas arquitecturas. En la Figura 29 se representan estas dos arquitecturas:

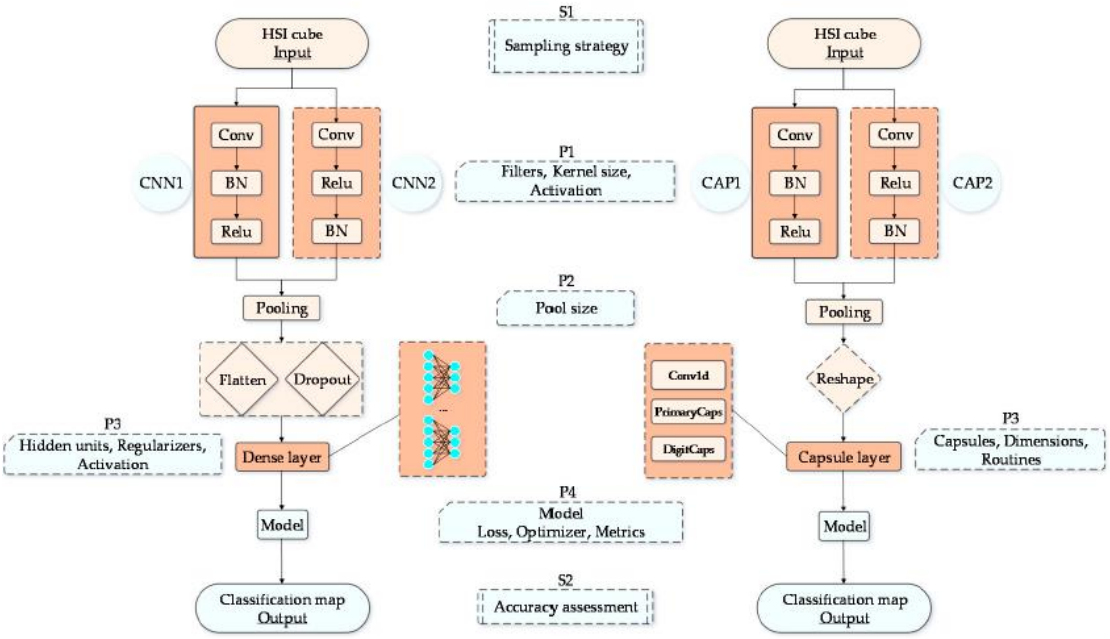

FIGURA 29. ARQUITECTURAS A COMPARAR CNN (IZDA) Y CAPSNET (DCHA)

Fuente: (F. Deng et al., 2018)

Las redes de cápsulas lograron mejor resultado que las redes convolucionales, consiguiendo las primeras 95.9% de precisión y 95.11% las segundas.

Por último, se presenta un artículo sobre clasificación de logos de vehículos (Chen et al., 2018). El reconocimiento del logotipo del vehículo es una parte importante de la identificación de vehículos en sistemas inteligentes de transporte. La tarea de reconocer el logotipo del coche ha estado dominada en los últimos años por las redes neuronales convolucionales. Sin embargo, las redes convolucionales no funcionan bien con imágenes rotadas, o bien cuando tienen mucho ruido, por citar dos inconvenientes. El artículo, propone un marco de trabajo para reconocimiento de imágenes de logotipo con una red de cápsulas. Se basa en que las redes cápsulas guardan la información en un vector cuyos parámetros de instanciación son, por ejemplo, las posiciones y las orientaciones de las imágenes y sus elementos, y en que el algoritmo empleado para entrenar la red se realiza mediante un enrutamiento dinámico, que es más efectivo que el de agrupamiento, o *max pooling,* de las redes convolucionales. Prueban en el estudio, con diferentes cambios de imagen, como la rotación y la oclusión. Y también se tienen en cuenta degradaciones de la imagen, como el ruido y el desenfoque.

De esta forma, el estudio ofrece una solución basada en redes de cápsulas para la clasificación de logotipos de coches donde la precisión llega al 100%, superando así a las redes convolucionales.

Resumen

Se ha hecho un repaso donde se ven tanto los buenos resultados que han dado las CNN, como los problemas que presentan. También se ha realizado un estudio sobre las redes de cápsulas, y el potencial que presentan, lo que las hace muy interesantes de estudiar. Las principales ventajas de las redes de cápsulas, frente a las redes convolucionales son las siguientes:

- No necesitan para el entrenamiento gran cantidad de datos (ni usar redes pre-entrenadas con esos datos).
- En la fase de entrenamiento no pierden información. Las redes convolucionales pierden mucha información en sus capas de pooling, que luego resuelven construyendo complejas arquitecturas para recuprar esa información perdida.
- Manejan bien las ambigüedades, debido a que guardan en un vector toda la información relativa a su estado espacial. Con las cápsulas, al mantener toda la información espacial relativa a los objetos, y las relaciones que hay entre ellos, esto deja de suceder. Y las CNN necesitan de componentes extra para poder identificar qué parte pertenece a qué objeto, mientras que las cápsulas al almacenar la jerarquía entre los objetos de la imagen y sus componentes con toda la información relativa a su posición, esto no sucede.

A pesar de todas estas ventajas que en principio presentan las cápsulas, éstas están todavía lejos de funcionar tan bien como las CNN en imágenes grandes como las de ImageNet, por ejemplo. También hay que decir que son computacionalmente muy costosas y que no pueden detectar dos objetos diferentes cuando estos están muy cerca (curiosamente, esto también sucede con las personas, Krizhevsky et al., 2012). Pero las ideas que presenta son muy prometedoras y pudiera ser que se esté al inicio de una nueva era en cuanto al reconocimiento de imágenes. Hay que recordar, que las CNN aparecieron por primera vez en 1998, pero que no fue hasta el 2012 cuando empezaron a dar unos resultados innovadores y excelentes. Y las cápsulas aparecieron en 2012, pero hasta 2017 no apareció el método para entrenarlas.

3. Objetivos y metodología del trabajo.

Objetivos

El principal objetivo de este trabajo es ver el desempeño que tienen las arquitecturas basadas en redes de cápsulas en la tarea de clasificación de imágenes en comparación con las redes convolucionales. Actualmente, estas últimas, mantienen desde hace casi una década el estado del arte con diferentes modelos de arquitecturas. Para lograr el objetivo de este estudio, se ha realizado una comparativa de dos redes neuronales, con arquitecturas distintas, en distintos conjuntos de datos, para ver qué resultado obtienen cada una al clasificar sobre los mismos *datasets*. Se han elaborado dos modelos de redes neuronales con arquitecturas diferentes y se han comparado los resultados. Para alcanzar el objetivo general ha sido necesario conseguir los siguientes objetivos específicos:

1. Revisión del estado del arte para identificar cuáles son las principales arquitecturas de redes neuronales.
2. Búsqueda y revisión de arquitecturas basadas en bloques residuales.
3. Búsqueda y revisión de arquitecturas basada en redes de cápsulas.
4. Búsqueda y estudio de las distintas herramientas, librerías y frameworks para el desarrollo de los algoritmos de las redes neuronales a seleccionar.
5. Estudio, revisión y modificación de los algoritmos seleccionados en código fuente.
6. Búsqueda y estudio de conjuntos de datos a utilizar con los algoritmos seleccionados.

Metodología de trabajo

Para lograr el objetivo de este trabajo se ha realizado una metodología basada en fases que consta de las siguientes etapas:

- Fase 1. Se realiza un amplio y profundo estudio y selección de los diferentes modelos existentes de redes neuronales a implementar y modificar. En esta fase se realiza una búsqueda bibliográfica de las distintas arquitecturas de redes neuronales habiéndonos centrado en las redes convolucionales y redes de cápsulas. Se consultaron artículos científicos, revistas y artículos web entre otras fuentes. Esto permitió finalmente seleccionar una arquitectura basada en ResNet (He et al., 2016a). Se escogieron tres modelos para poder ver las diferencias según la profundidad de la red. Estos modelos han sido ResNet18 (He et al., 2016a), ResNet34(He et al., 2016a) y ResNet50(He et al., 2016a). Para el modelo de redes

de cápsulas se eligió el modelo basado en su estudio original (G. E. Hinton et al., 2011).

- Fase 2. Selección de los *dataset* de pruebas. En esta fase se ha hecho una búsqueda y un estudio detallado de diferentes *datasets* para ser estudiados. Se han buscado *datasets* de diferentes complejidades para ver los resultados en los modelos de redes en los diferentes conjuntos.
- Fase 3. Desarrollo e implementación de los algoritmos. En esta fase se recogen los algoritmos seleccionados y se adapta el código Python para poder ejecutarlos sobre los distintos *datasets*. Se realizan las mejoras y ajustes necesarios también para poder leer y ajustar los *datasets* seleccionados.
- Fase 4. Ejecución y estudio de los resultados. En esta etapa se ejecutan todos los modelos con los distintos conjuntos de datos. Para el conjunto de datos se realizaron dos ejecuciones: una con la técnica de *data augmentation* y otra sin ella. También se realizó la comparación de los resultados. De cada conjunto de datos se toman datos de entrenamiento y datos de validación para determinar la eficiencia y tiempos necesitados por ambas arquitecturas.

4. Análisis de arquitecturas y DataSets.

El análisis comparativo que se ha realizado entre las arquitecturas seleccionadas se ha realizado probando los modelos en cuatro *datasets* y eligiendo varios modelos de redes. A continuación, se explica con detalle qué *datasets* y qué arquitecturas se han elegido y porqué.

4.1 DataSet

Para poder evaluar las arquitecturas elegidas se han utilizado cuatro conjuntos de datos de diferentes características, cantidad de datos y complejidad. Los conjuntos elegidos han sido, primero Mnist (LeCun & Cortes, MNIST handwritten digit database, 2010) por ser uno de los conjuntos más estudiados, simples y ligeros que existen. De esta forma se tiene una manera relativamente sencilla y rápida de ver que la arquitectura del modelo seleccionado funciona. Luego, se ha seleccionado Fashion-Mnist (Guo, 2017, p.) por ser también bastante simple, pero con imágenes algo más complejas que Mnist (LeCun & Cortes, 2010). Finalmente, como conjuntos más complejos y con imágenes reales se han seleccionado COIL-100 (Nene et al., 1996) y Cifar10 (Krizhevsky et al., s. f., p. 10), también por ser conjuntos muy probados en todo tipo de arquitecturas, siendo estas imágenes más complicadas, ya que son imágenes reales.

A continuación, se van a detallar los *datasets* seleccionados para realizar las pruebas, mostrando sus principales características y una pequeña muestra de cada uno de ellos.

Mnist (LeCun & Cortes, 2010) es un conjunto de datos compuesto de imágenes de números manuscritos del 0 al 9. Está formado por 60000 imágenes de entrenamiento y 10000 de validación. Son imágenes en blanco y negro, con un tamaño de 28x28, normalizadas y los números están centrados en la imagen. Se ha usado porque es un conjunto sencillo, ligero, grande y ha sido muy estudiado en todo tipo de arquitecturas de redes neuronales. En la se puede ver un ejemplo del *dataset* Mnist (LeCun & Cortes, 2010).

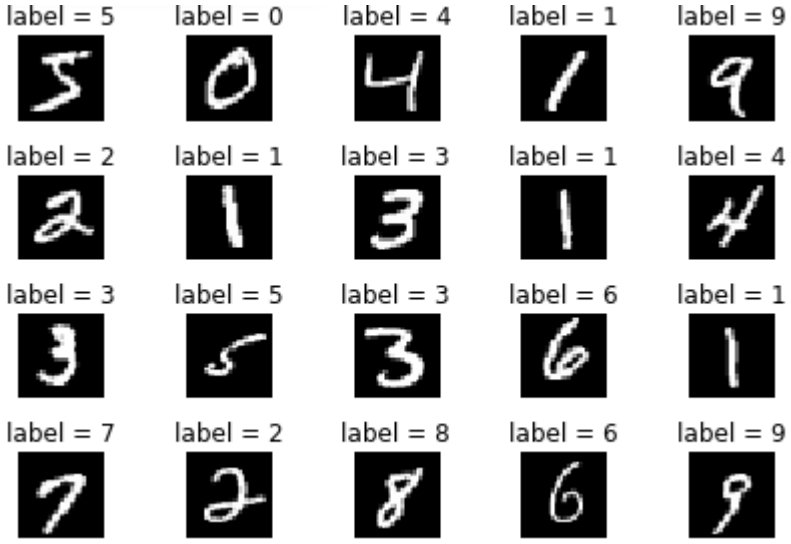

FIGURA 30. EJEMPLO DEL DATASET DE MNIST

Fuente: Elaboración propia

Fashion-Mnist (Xiao et al., 2017) es un conjunto de imágenes de artículos de vestir de Zalando [1] que contiene 60.000 imágenes de ejemplo para entrenar y un conjunto de test formado por 10.000 imágenes más. Las imágenes tienen un tamaño de 28x28 pixels, son en blanco y negro y están distribuidas en 10 clases diferentes. Este conjunto de datos tiene una estructura muy similar a Mnist (LeCun & Cortes, 2010) y está hecho así para ahorrar trabajo cuando se prueben en los mismos algoritmos. Se ha elegido este conjunto por ser algo más complicado que Mnist (LeCun & Cortes, 2010), pero sigue siendo simple ya que las imágenes son siluetas de artículos de vestir como se puede ver en la . En esta Figura 31 se puede ver varios ejemplos de las diez clases del *dataset* Fashion-Mnist (Xiao et al., 2017), donde cada clase tiene tres filas.

[1] https://www.zalando.es/

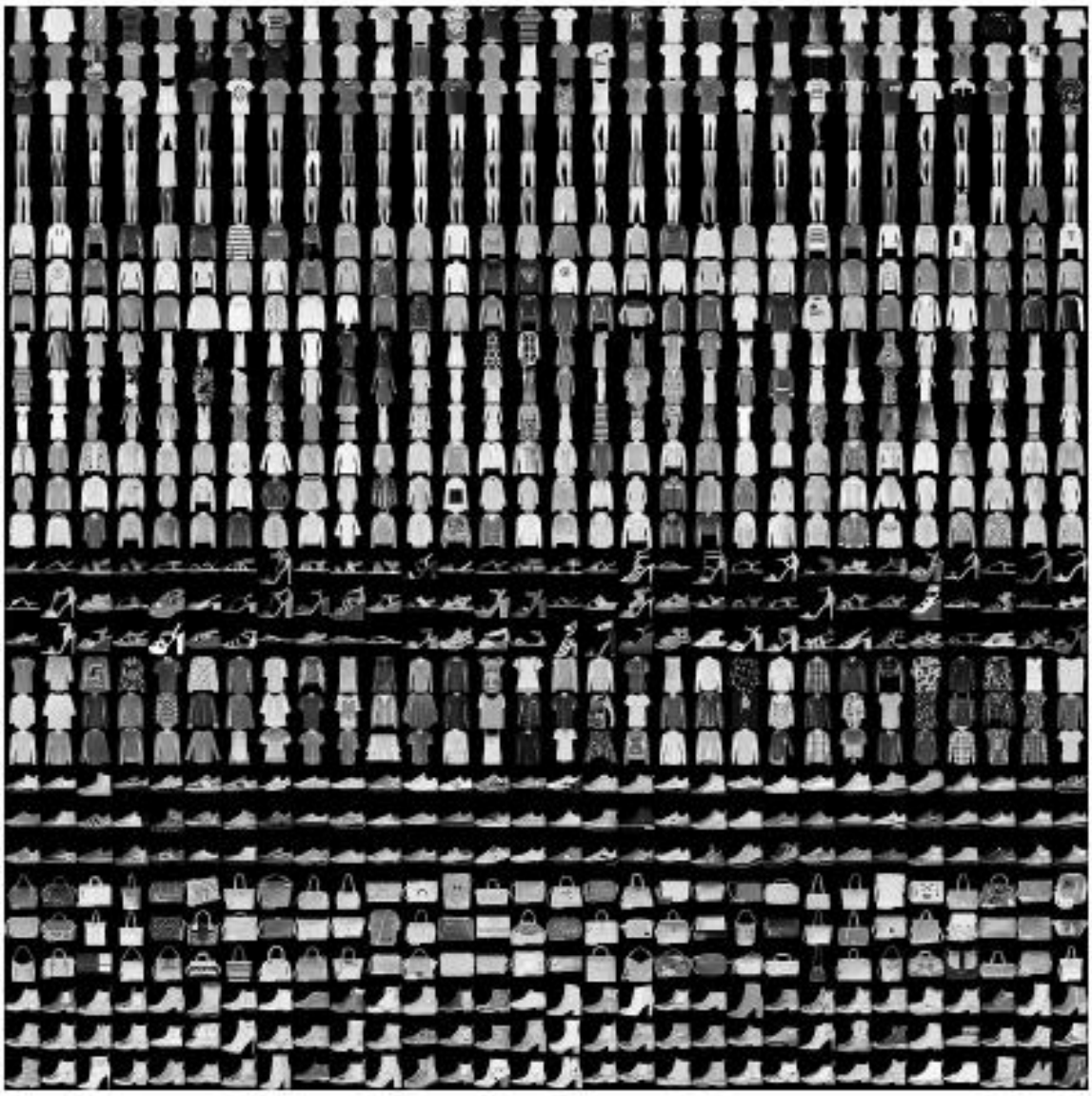

FIGURA 31. EJEMPLO DEL DATASET FASHION MNIST

Fuente: (Xiao et al., 2017)

CIFAR-10 (Krizhevsky et al., s. f.) es un conjunto de imágenes mucho más complicado que el anterior. Son imágenes reales que tienen su fondo correspondiente, es decir no están aisladas. Está formado por 60.000 imágenes divididas en 10 categorías, con un tamaño de la imagen de 32x32. Está dividido en 50.000 imágenes de entrenamiento y 10.000 imágenes de test. Se usará este conjunto de datos por ser un conjunto de imágenes reales, lo que le añade bastante complejidad a la hora de clasificar en categorías. Es un conjunto amplio, con imágenes muy variadas, y también ha sido usado en innumerables estudios de todo tipo de arquitecturas de redes neuronales. De los cuatro *dataset* elegidos, se puede decir que este es el más complicado, al ser las imágenes reales, no estar aisladas y por tener un gran número

de ejemplos para entrenar. En la Figura 32 se pueden ver diez ejemplos de las diez categorías (a la izquierda) de este conjunto de datos.

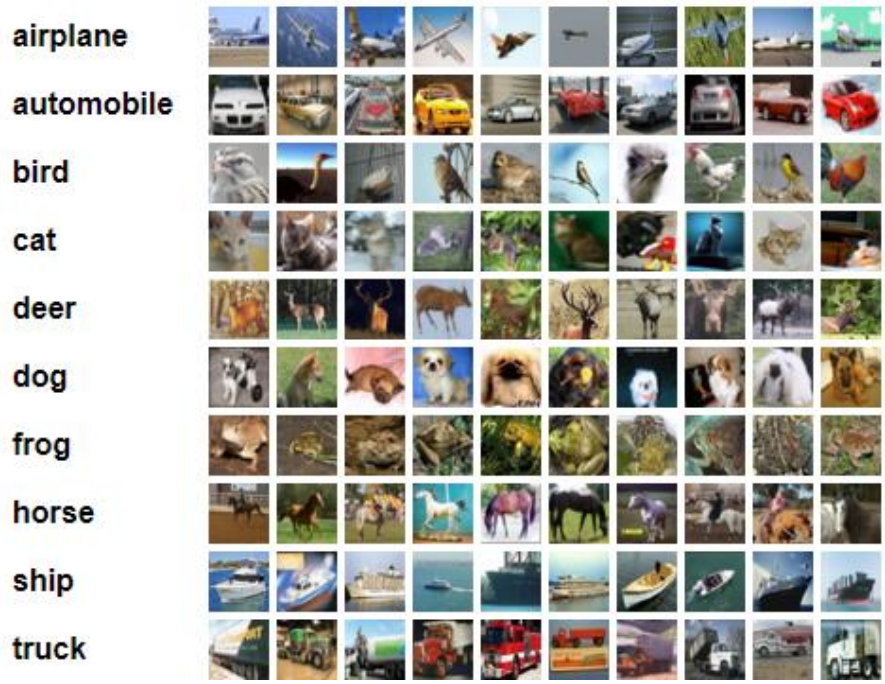

FIGURA 32. EJEMPLOS DE LAS CLASES DE CIFAR-10 EN CADA UNA DE SUS CATEGORÍAS

Fuente: (Krizhevsky et al., s. f.)

COIL-100 (Nene et al., 1996). Este es un conjunto de datos formado por 100 objetos reales distintos, y de cada objeto hay 72 poses en distintos ángulos (con cinco grados de diferencia en cada imagen). Cada objeto es una categoría y los ejemplos son el mismo objeto rotado con los distintos ángulos. Se ha seleccionado este conjunto porque tiene muy pocos ejemplos, y el *dataset* está compuesto por variaciones de cada clase. Precisamente una de las fortalezas teóricas que presentan las redes de cápsulas es su capacidad para almacenar los datos relativos a la pose como la orientación o la posición, mientras que las CNN no, y en este tipo de imágenes es frecuente encontrar falsos negativos, y también, que precisan menos datos para los entrenamientos que las redes convolucionales. En la Figura 33 se pueden ver ejemplos de distintos objetos del conjunto.

FIGURA 33. EJEMPLOS DE IMÁGENES DE COIL-100.

(Fuente: Elaboración propia)

4.2 Implementación de soluciones

En esta etapa, se ha seleccionado y modificado en Python[2] los algoritmos para las redes neuronales que se han elegido. El desarrollo se ha hecho en Python y se ha usado la librería de redes neuronales de código abierto Keras [3], la cual se ha ejecutado sobre TensorFlow[4]. Se ha modificado y adaptado el código para que procese los datos, los guarde y luego los muestre en los cuatro conjuntos de datos seleccionados. Por cada conjunto de datos se ha

[2] https://www.python.org/
[3] https://keras.io/
[4] https://www.tensorflow.org/

entrenado cada modelo dos veces, una vez con técnica de *data augmentation* (DA) y otra sin ella. Esto se ha hecho así para ver cuál es el efecto sobre las redes de cápsulas, ya que es una manera de aumentar el número de ejemplos para entrenar, y ver cómo afectan a ambas arquitecturas, sobre todo a las cápsulas, ya que éstas en la fase de *autoencoder* ellas mismas generan imágenes distorsionadas de la original que están estudiando.

También se ha usado una función para que una vez que la red no aprenda más, el entrenamiento se detenga. Esta función se llama a modo de callback, y en Keras se denomina *EarlyStopping.* Se le pasa un parámetro que indica a partir de cuántas épocas sin mejorar los resultados se tiene que detener, este valor está fijado por defecto a 10. Se ha usado esta función para intentar que los entrenamientos sean más cortos en el tiempo.

4.2.1 CNN. Red neuronal basada en una red neuronal convolucional

Para la red neuronal basada en CNN, se ha elegido la arquitectura ResNet (He et al., 2016a) por ser una arquitectura muy estudiada y probada en la tarea de clasificación de imágenes. Se puede decir que este modelo es todo un clásico y referente en las distintas tareas de visión por computador, entre ellas, estudiar la clasificación de imágenes. Para el desarrollo de nuestros modelos, se ha elegido una solución alojada en GitHub, basada en código de (Kotikalapudi, 2017). El código consiste en tres modelos con distintas profundidades de capas: ResNet18 (He et al., 2016a), ResNet34 (LeCun & Cortes, 2010) (He et al., 2016a)y ResNet50 (He et al., 2016a).

El código ejecuta una función principal, que por cada ejecución entrena un modelo con un conjunto de datos. Cada ejecución llama a una función que creará y entrenará el modelo correspondiente con su conjunto de datos que se haya seleccionado.

Para cada modelo (Coil100, Fashion, Mnist (LeCun & Cortes, 2010) y Cifar10 (Krizhevsky et al., s. f., p. 10)) se ha construido una función llamada train+dataset. Esta función, en base a los parámetros que se le van pasando crea y entrena cada modelo.

A continuación, se presenta una parte del código Python para Cifar10 (Krizhevsky et al., s. f., p. 10):

```
if int(sys.argv[1]) == 2: #Cifar10
        if int(sys.argv[2]) == 18 or int(sys.argv[2]) == 34 or int(sys.argv[2]) == 50:
                print('trainCifar10')
                trainCifar10(sys.argv[2])
```

Luego, la función trainCifar10(res_model), dependiendo del parámetro res_model, creará el model ResNet18 (He et al., 2016a), ResNet34 (He et al., 2016a) o bien ResNet50 (He et al., 2016a):

```
if int(res_model) == 18:
        model_r18 = resnet.ResnetBuilder.build_resnet_18((img_channels, img_rows, img_cols), nb_classes)
        nombre = 'cifar10_resnet18'
        train(model_r18, nombre, data)
```

Una vez construido el modelo, se entrena, una vez con *data augmentation*, otra vez sin ella.

A continuación, en la se puede ver la arqutiectura de capas y el número de parámetros de cada modelo utilizado. En la parte superior de la imagen se ve el bloque residual base. Este bloque, que es el que luego se repetirá para formar la arquitectura, está formado por una capa convolucional, una capa de normalización, una función de activación y finalmente una capa de *max pooling*. Este bloque base es el que se repite a lo largo de las arquitecturas 17 veces para ResNet18 (He et al., 2016a), 33 veces para ResNet34 (He et al., 2016a) y 49 veces para ResNet50 (He et al., 2016a). Hay que tener en cuenta que a este número de capas convolucionales hay que sumarle la capa de entrada, de ahí que se multiplique por un número inferior al que lleva en el nombre. Dado que son arquitecturas muy profundas, para representar los modelos en la Figura 34 se han esquematizado, presentando en un primer lugar la capa convolucional inicial, seguido del primer bloque base ,y terminando con el último bloque de cada modelo, indicando también el número de paramátros totales.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 1)	0	
conv2d_1 (Conv2D)	(None, 14, 14, 64)	3200	input_1[0][0]
batch_normalization_1 (BatchNor	(None, 14, 14, 64)	256	conv2d_1[0][0]
activation_1 (Activation)	(None, 14, 14, 64)	0	batch_normalization_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0	activation_1[0][0]

x 17, 33, 49

ResNet 18

batch_normalization_17 (BatchNo	(None, 1, 1, 512)	2048	add_8[0][0]
activation_17 (Activation)	(None, 1, 1, 512)	0	batch_normalization_17[0][0]
average_pooling2d_1 (AveragePoo	(None, 1, 1, 512)	0	activation_17[0][0]
flatten_1 (Flatten)	(None, 512)	0	average_pooling2d_1[0][0]
dense_1 (Dense)	(None, 10)	5130	flatten_1[0][0]

Total params: 11,186,186
Trainable params: 11,178,378
Non-trainable params: 7,808

ResNet34

batch_normalization_33 (BatchNo	(None, 1, 1, 512)	2048	add_16[0][0]
activation_33 (Activation)	(None, 1, 1, 512)	0	batch_normalization_33[0][0]
average_pooling2d_1 (AveragePoo	(None, 1, 1, 512)	0	activation_33[0][0]
flatten_1 (Flatten)	(None, 512)	0	average_pooling2d_1[0][0]
dense_1 (Dense)	(None, 10)	5130	flatten_1[0][0]

Total params: 21,305,482
Trainable params: 21,290,250
Non-trainable params: 15,232

ResNet50

batch_normalization_49 (BatchNo	(None, 1, 1, 2048)	8192	add_16[0][0]
activation_49 (Activation)	(None, 1, 1, 2048)	0	batch_normalization_49[0][0]
average_pooling2d_1 (AveragePoo	(None, 1, 1, 2048)	0	activation_49[0][0]
flatten_1 (Flatten)	(None, 2048)	0	average_pooling2d_1[0][0]
dense_1 (Dense)	(None, 10)	20490	flatten_1[0][0]

Total params: 23,586,570
Trainable params: 23,541,130
Non-trainable params: 45,440

FIGURA 34. ARQUITECTURAS RESNET

Fuente: Elaboración propia

El código fuente utilizado está alojado en GitHub[5].

[5] https://github.com/lara019/Convolutional-Residual-Network

4.2.2 CapsNet. Red neuronal que sigue una arquitectura basada en redes de cápsulas.

Se ha elegido esta arquitectura porque la teoría muestra que estas redes podrían superar a las CNN ya que no tienen los problemas o carencias que presentan las primeras. Esta arquitectura, por ser muy novedosa no existen los numerosos estudios y diseños que sí existen en las redes convolucionales. El modelo base para la solución buscada es de nuevo una solución alojada en GitHub (Guo, 2017), usando Python con la librería de Keras sobre TensorFlow. Se han hecho las modificaciones correspondientes para que el código soporte los cuatro conjuntos de datos, y poder mostrar los resultados.

En la Figura 35 se puede ver la arquitectura que se ha usado.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 28, 28, 1)	0	
conv1 (Conv2D)	(None, 20, 20, 256)	20992	input_1[0][0]
primarycap_conv2d (Conv2D)	(None, 6, 6, 256)	5308672	conv1[0][0]
primarycap_reshape (Reshape)	(None, 1152, 8)	0	primarycap_conv2d[0][0]
primarycap_squash (Lambda)	(None, 1152, 8)	0	primarycap_reshape[0][0]
digitcaps (CapsuleLayer)	(None, 10, 16)	1474560	primarycap_squash[0][0]
input_2 (InputLayer)	(None, 10)	0	
mask_1 (Mask)	(None, 160)	0	digitcaps[0][0] input_2[0][0]
capsnet (Length)	(None, 10)	0	digitcaps[0][0]
decoder (Sequential)	(None, 28, 28, 1)	1411344	mask_1[0][0]

Total params: 8,215,568
Trainable params: 8,215,568
Non-trainable params: 0

FIGURA 35. ARQUITECTURA CAPSNET

Fuente: Elaboración propia

También se pueden ver las distintas capas de esta arquitectura: la primera capa convolucional, la capa PrimaryCaps, la capa DigitCaps, formando lo que se ha llamado el *encoder,* y finalmente el *decoder.*

La parte del *encoder* se codificaría de la siguiente manera:

La imagen de entrada:

```
x = layers.Input(shape=input_shape)
```

La capa convolucional:

```
conv1 = layers.Conv2D(filters=256, kernel_size=9, strides=1, padding='valid',
activation='relu', name='conv1')(x)
```

La capa PrimaryCaps: una capa convolucional 2D, con función de activación *squash* para luego dejarla de un tamaño (none, num_capsule, num_capsule)

```
primarycaps = PrimaryCap(conv1, dim_capsule=8, n_channels=32, kernel_size=9, strides=2,
padding='valid')
```

La capa DigitCaps: donde el algoritmo de enrutamiento hace su función.

```
digitcaps = CapsuleLayer(num_capsule=n_class, dim_capsule=16,
num_routing=num_routing,

                                    name='digitcaps')(primarycaps)
```

Y finalmente la parte del *decoder,* con las tres capas *fully connected*:

```
decoder = models.Sequential(name='decoder')

decoder.add(layers.Dense(512, activation='relu', input_dim=16*n_class))

decoder.add(layers.Dense(1024, activation='relu'))

decoder.add(layers.Dense(np.prod(input_shape), activation='sigmoid'))

decoder.add(layers.Reshape(target_shape=input_shape, name='out_recon'))
```

El código fuente utilizado está alojado en GitHub[6].

4.3 Análisis de los resultados

Una vez entrenados los cuatro modelos, tanto con la técnica de *data augmentation* (DA) como sin ella, se presentan los resultados.

En primer lugar, se muestran las tablas con los resultados, para pasar seguidamente a las conclusiones.

	Parámetros
CapsuleNetwork	8.215.568
ResNet18	11.186.186 (7.808)

[6] https://github.com/lara019/CapsuleNetwork

ResNet34	21.305.482 (15.232)
ResNet50	23.586.570 (45.440)

TABLA 1. PARÁMETROS DE LAS DISTINTAS ARQUITECTURAS

En la se puede ver el número de parámetros a entrenar (superior), y parámetros que no se entrenan en el número inferior. Destaca la gran diferencia de parámetros a entrenar entre las distintas arquitecturas, teniendo CapsNet casi un 27% menos de parámetros comparada con ResNet18 (He et al., 2016a) que es la más simple, y un 35% menos comparada con ResNet50 (He et al., 2016a), que es la más profunda. No obstante, esta reducción de parámetros no se corresponde con una reducción en el tiempo de entrenamiento como se verá en la siguiente tabla.

Tiempo /DA	Fashion-mnist	Mnist	Cifar10	Coil100
CapsuleNetwork	53 min/ 24 ep 127 min/ 56 ep	63.5 min/ 30 ep 81 min / 38 ep	57 min/ 32 ep 143 min / 75 ep	166 min/ 95 ep 116 min / 70 ep
ResNet18	24 min/ 21 ep 98 min / 74 ep	42 min/ 46 ep 22 min / 22 ep	20 min/ 16 ep 55 min / 37 ep	8 min/ 52 ep 20 min / 73 ep
ResNet34	41 min/ 20 ep 119 min / 56 ep	85 min/ 53 ep 39 min / 23 ep	36 min/ 17 ep 83 min / 35 ep	19 min/ 77 ep 21 min / 63 ep
ResNet50	75 min/ 33 ep 163 min / 69 ep	149 min/ 63 ep 82 min / 33 ep	83.7 min/ 27 ep 180 min / 72 ep	41 min/ 103 ep 12 min / 27 ep

TABLA 2. TIEMPO/EPOCHS

En la se puede ver el tiempo empleado en cada entrenamiento de cada modelo con un conjunto de datos. El número superior se corresponde con un entrenamiento sin *data augmentation* (DA), y en el inferior sí se ha usado esta técnica.

En casi todos los casos, se ve que el tiempo utilizado con el entrenamiento con DA ha sido muy superior a sin ella, a excepción de CapsNet con Coil100 y de ResNet34 (He et al., 2016a) con Mnist (LeCun & Cortes, 2010).

Acc/loss	Fashion-mnist	Mnist	Cifar10	Coil100

CapsuleNetwork	0.9910 / 0.0149 0.9393 / 0.0523	1.0 / 0.035 0.9949 / 0.0118	0.8499 / 0.1647 0.7143/0.2217	0.9913 / 0.0309 0.9778 / 0.0464
ResNet18	0.9535/0.2573 0.9533/0.1939	1.0 / 0.0138 0.9873 / 0.0537	0.9802 / 0.3418 0.9459 / 0.3521	1.0 / 0.0619 0.998 / 0.0366
ResNet34	0.9834/0.1487 0.9520/0.1989	1 / 0.0155 0.9874 / 0.0562	0.9799 / 0.3427 0.9497 / 0.3429	1 / 0.0492 1 / 0.0427
ResNet50	0.997/0.1061 0.9374/0.2430	0.9995 / 0.0224 0.9952 / 0.0452	0.9961 / 0.2802 0.9389 / 0.3472	1 / 0.0442 0.9996 / 0.0474

TABLA 3. ACCURACY/ERROR

En la se tienen los resultados del entrenamiento en forma de precisión/error. Los valores superiores se corresponden con el entrenamiento sin DA, mientras que los inferiores se corresponden a un entrenamiento donde sí se ha usado esta técnica.

Val_Acc/_Val_loss	Fashion-mnist	Mnist	Cifar10	Coil100
CapsuleNetwork	0.9103 / 0.0781 0.9230 / 0.0639	0.9945 / 0.0099 0.9895 / 0.0139	0.5578/0.3225 0.6818/0.2350	0.9660/0.0488 0.9688/0.0114
ResNet18	0.8952/0.4845 0.9248/0.2869	0.9934 / 0.0521 0.9905 / 0.0452	0.7947 / 1.2733 0.8603 / 0.6846	1 / 0.0608 1 / 0.0359
ResNet34	0.9061/0.5014 0.9281/0.2863	0.9930 / 0.0540 0.9901 / 0.0513	0.7943 / 1.2808 0.8628 / 0.6820	1 / 0.0486 1 / 0.0417
ResNet50	0.9079/0.6308 0.9258/0.2862	0.9933 / 0.0671 0.9908 / 0.0635	0.7795 / 1.5302 0.8724 / 0.5932	1 / 0.0440 0.9938 / 0.0659

TABLA 4. VALIDATION ACCURACCY/VALIDATION LOSS

En la están los valores de los datos de validación de los modelos. Tiene el mismo formato que la tabla anterior, precisión/error, en la fila inferior con data augmentation y el la superior si ella.

Para realizar la comparación de las redes de cápsulas con la red convolucional, se van a coger los conjuntos que mejores resultados hayan tenido comparando si se ha usado la técnica de data augmentation (DA) o si no se ha usado (SIN DA), indicando en la celda el resultado que se ha dejado. Si la celda tiene un * es que los resultados han sido muy similares, y se ha escogido el que ha tardado menos tiempo en el entrenamiento. En caso que ponga *Overfitting* significa que los valores del modelo en fase de validación han sido mejores que los valores en fase de entrenamiento, en cuyo caso no se da por válido el modelo para el *dataset*, al habérselo *aprendido de memoria*. De esta forma, el modelo en futuras predicciones con ejemplos distintos a los del *dataset* dará errores muy superiores a los conseguidos en las fases de entrenamiento y validación.

Acc/loss	Fashion-mnist	Mnist	Cifar10	Coil100
CapsuleNetwork	0.9910 / 0.0149 SIN DA	0.9949 / 0.0118 DA	0.8499 / 0.1647 SIN DA	0.9913 / 0.0309 SIN DA
ResNet18	0.9533/0.1939 DA	1.0 / 0.0138 SIN DA	0.9802 / 0.3418 SIN DA	Overfitting
ResNet34	0.9834/0.1487 SIN DA	1 / 0.0155 SIN DA	0.9799 / 0.3427 SIN DA	Overfitting
ResNet50	0.997/0.1061 SIN DA	0.9995 / 0.0224 SIN DA	0.9961 / 0.2802 SIN DA	Overfitting

TABLA 5 MEJORES RESULTADOS ENTRENAMIENTO (CON O SIN DA)

La , muestra que en la fase de entrenamiento casi todos los resultados son mejores cuando se no se ha usado DA.

Sin embargo, si se mira en la tabla de los datos de validación, nos queda la siguiente donde se han puesto los mejores resultados obtenidos. En este caso, hay una mayoría de resultados con DA.

Val_Acc/_Val_loss	Fashion-mnist	Mnist	Cifar10	Coil100
CapsuleNetwork	0.9230 / 0.0639 DA	0.9945 / 0.0099	0.6818/0.2350 DA	0.9688/0.0114 DA

		SIN DA		
ResNet18	0.9248/0.2869 DA	0.9905 / 0.0452 SIN DA*	0.8603 / 0.6846 DA	Overfitting
ResNet34	0.9281/0.2863 DA	0.9901 / 0.0513 DA*	0.8628 / 0.6820 DA	Overfitting
ResNet50	0.9258/0.2862 DA	0.9908 / 0.0635 DA*	0.8724 / 0.5932 DA	Overfitting

TABLA 6. MEJORES RESULTADOS VALIDACIÓN (SIN O CON DA)

En la , se puede cómo para el modelo de redes de cápsulas, los resultados han sido mejores usando DA, a excepción de Mnist (LeCun & Cortes, 2010). En los modelos de CNN también el resultado ha sido mejor usando DA en todos menos en ResNet18 (He et al., 2016a), que dependiendo del *dataset* o bien han sido mejores con DA, o bien el resultado ha estado igualado, pero ha sido más rápido en entrenamiento sin DA.

Si se realiza un análisis con respecto al *dataset* en los datos de validación, se tiene que los resultados han sido muy similares, es decir las precisiones han sido muy similares por conjuntos de datos.

Para Fasion-mnist todos los modelos han conseguido algo más de 92% de precisión con un error de 0.28, a excepción de las cápsulas que el error ha sido mucho más bajo. Y para todos los casos, el resultado ha sido mejor con DA. Para este *dataset*, el mejor resultado ha sido para ResNet50 (He et al., 2016a), siguiéndole las cápsulas de redes.

Para Mnist (LeCun & Cortes, 2010) la precisión ha sido superior al 99%, y las cápsulas han sido las que mejor resultado han obtenido. En este caso, los dos modelos que se podrían considerar más simples, ResNet18 (He et al., 2016a)y las Cápsulas han obtenido mejor resultado sin DA, mientras que ResNet34 (He et al., 2016a) y ResNet50 (He et al., 2016a), siendo más profundos, han necesitado de DA (más datos) para obtener mejores resultados.

Para Cifar10 (Krizhevsky et al., s. f., p. 10), estos resultados ya cambian. Las redes convolucionales han obtenido resultados sobre el 86% de precisión, frente a un 68% de las cápsulas. Y todos los mejores resultados han usado DA. Este conjunto al estar formado por imágenes reales, es el más complejo, y es el que peores resultados en general ha obtenido

comparado con los demás conjuntos. Y aquí se podría hacer notar que las redes convolucionales llevan una década siendo estudiadas y probadas, mientras que las redes de cápsulas apenas llevan dos años, con lo que es muy posible que a futuro tengan más recorrido.

Para Coil100 el único modelo que ha funcionado ha sido el de las redes de cápsulas. En los modelos ResNet (He et al., 2016a), estos no han conseguido converger y tienen overfitting (mejores valores en validación que en entrenamiento). Hay que destacar, que precisamente Coil100 es el conjunto que menos datos tiene. A pesar de ello, tiene 100 clases (número muy superior a las 10 clases de los otros conjuntos), y las imágenes (72 por cada una) son rotaciones de una misma. Es decir, es un conjunto para el que las redes de cápsulas, según la teoría, que nos dice que necesitan de menos datos para entrenar y guardan toda la información relativa a la posición, funcionarían mejor. Y en este caso, así ha sido. Hay que destacar también, que el tiempo empleado en el entrenamiento ha sido muy elevado.

Se va a realizar ahora un análisis por modelo, mirando las gráficas de los resultados de los *datasets*.

Para el conjunto de datos Cifar10 (Krizhevsky et al., s. f., p. 10), las gráficas de la Figura 36 y Figura 37, se ve que el mejor resultado en cuanto a precisión se refiere lo tiene las CNN Resnet18 (He et al., 2016a) y Resnet34 (He et al., 2016a), siendo además, de los que menos épocas necesitan para converger. En este caso, el modelo de cápsulas es el que peor resultado con diferencia ha tenido (aunque su tasa de error haya sido la más baja).

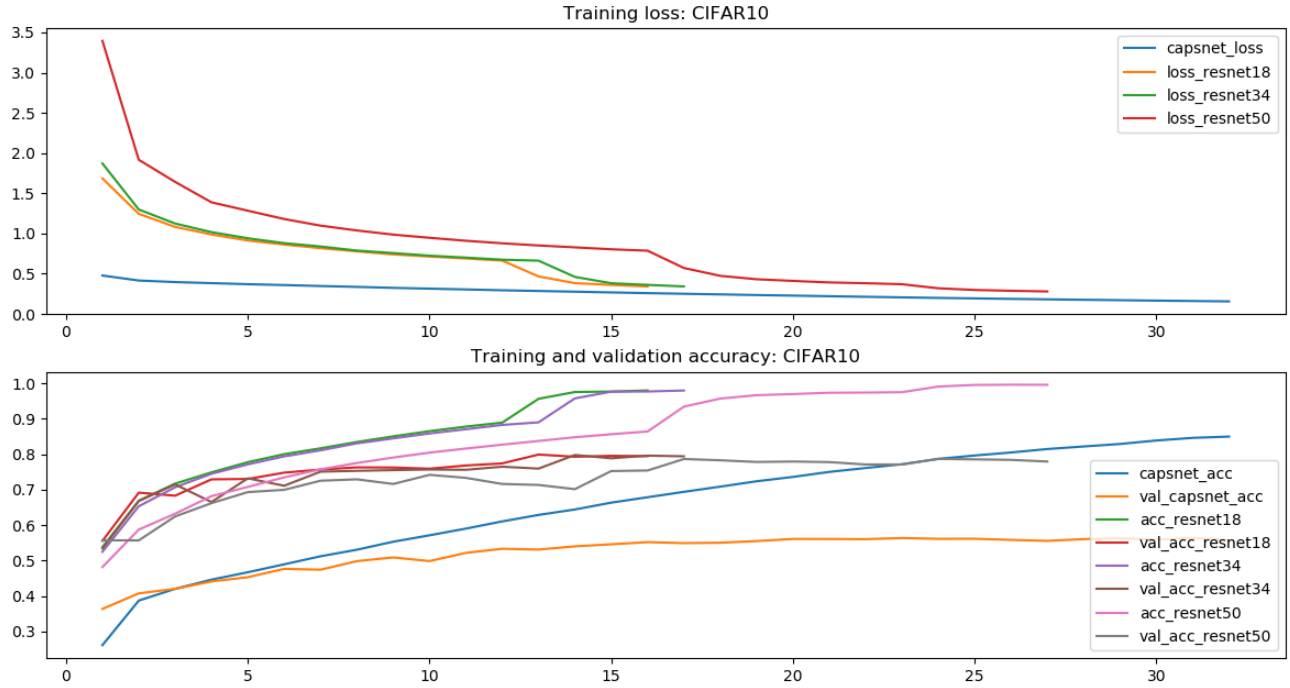

FIGURA 36. RESULTADOS DEL DATASET CIFAR10

Fuente: Elaboración propia

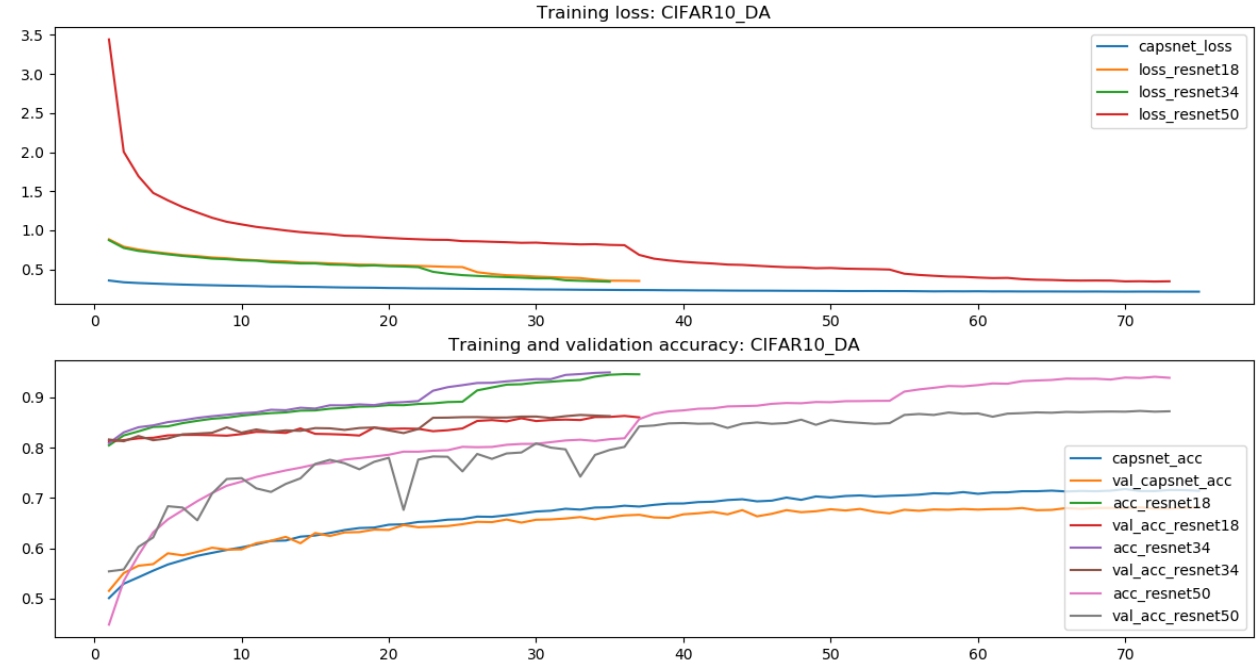

FIGURA 37. RESULTADOS DEL DATASET CIFAR10 CON *DATAAUGMENTATION*

Fuente: Elaboración propia

Para el caso del conjunto de datos, Fashion-Mnist (Guo, 2017, p.), en las figuras Figura 38 y Figura 39, se tiene que el mejor resultado ha sido para la arquitectura CNN, Resnet50 (He

et al., 2016a). Las redes de cápsulas, se han quedado muy cerca de este resultado, pero a su favor, han tardado casi un 50% menos del tiempo.

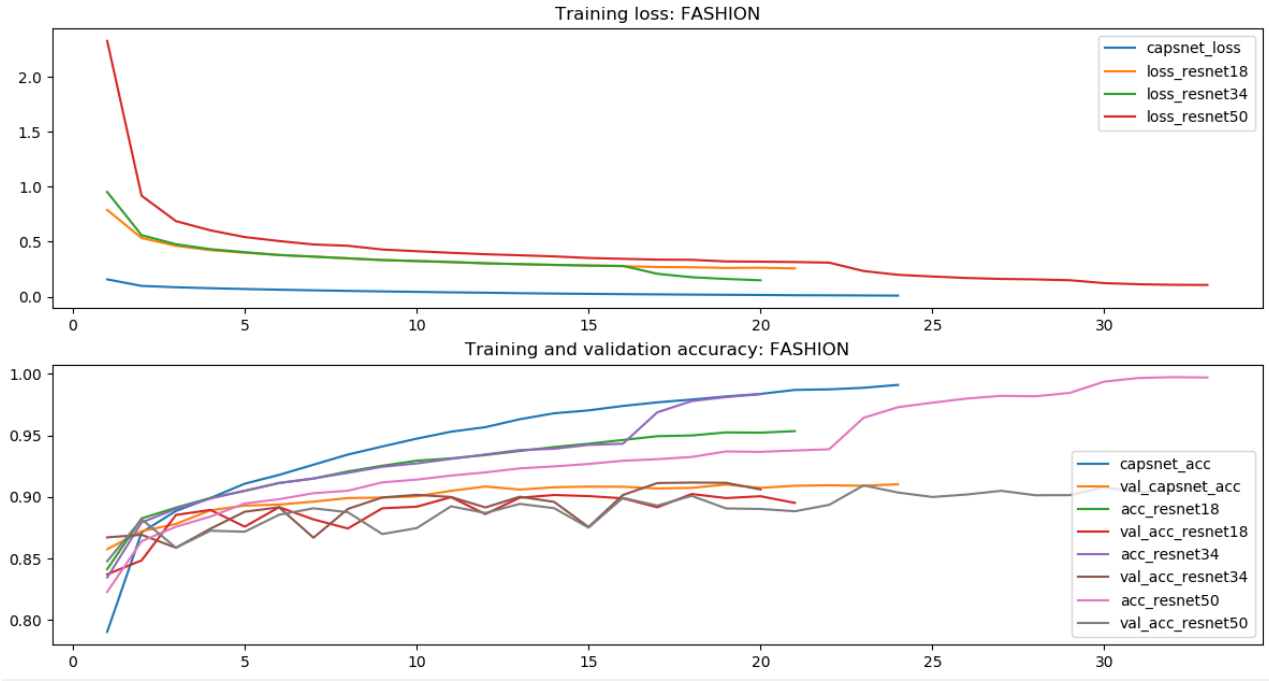

FIGURA 38. RESULTADOS DEL DATASET FASHION-MNIST

Fuente: Elaboración propia

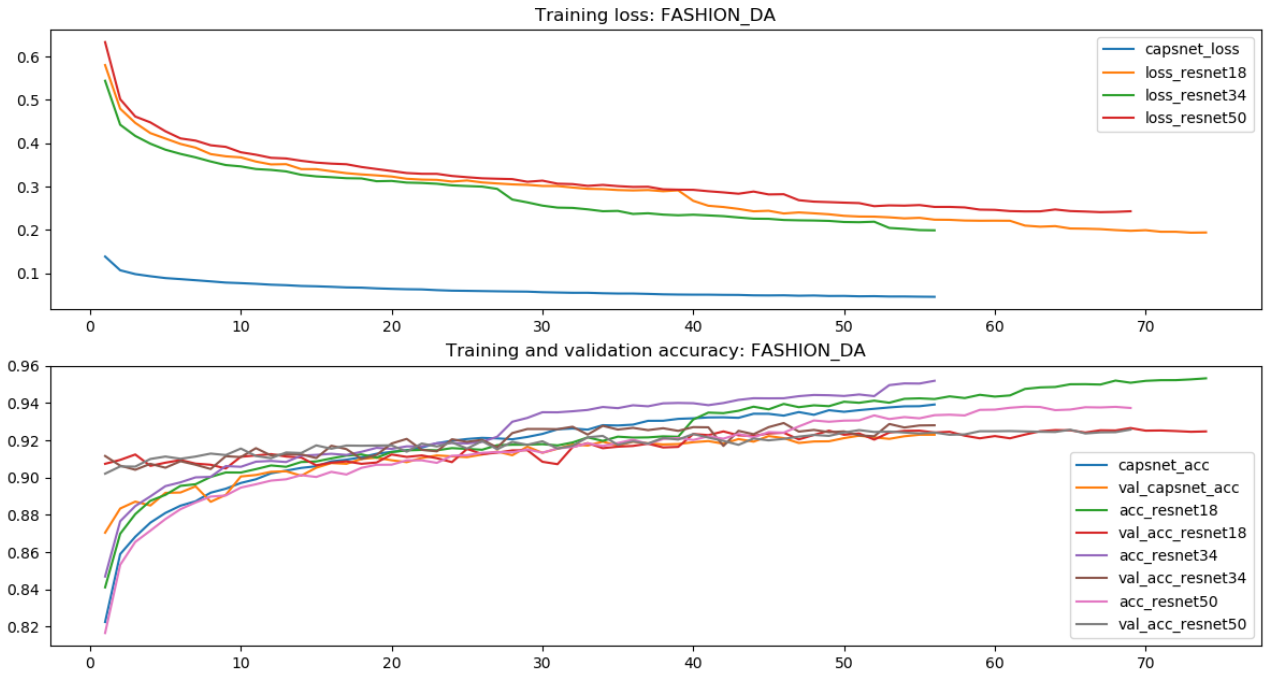

FIGURA 39.RESULTADOS DEL DATASET FASHION-MNIST CON *DATAAUGMENTATION*

En el caso de Coil100, en las Figura 40 y Figura 41, se ve que los resultados de validación en las arquitecturas ResNet (He et al., 2016a) tienen muchos picos. Esto es debido al *overfitting*. La red se ha aprendido de memoria los datos, y no es capaz de aprender de manera uniforme.

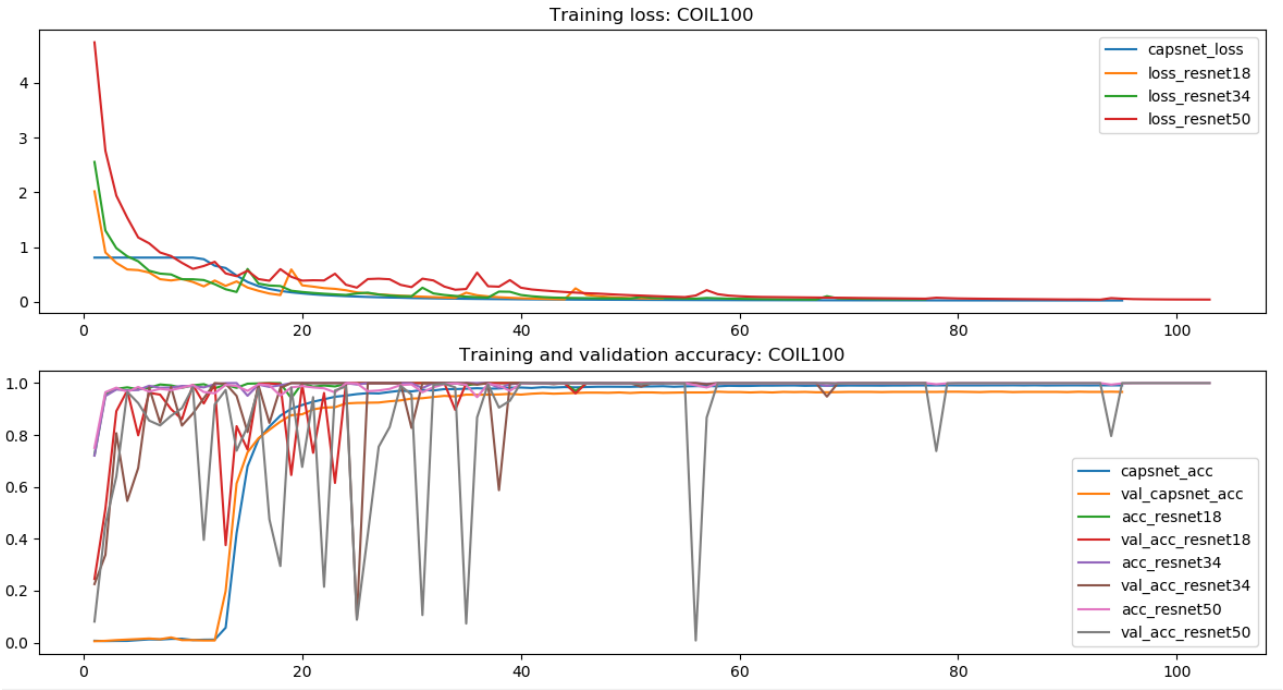

FIGURA 40. RESULTADOS DEL DATASET COIL-100

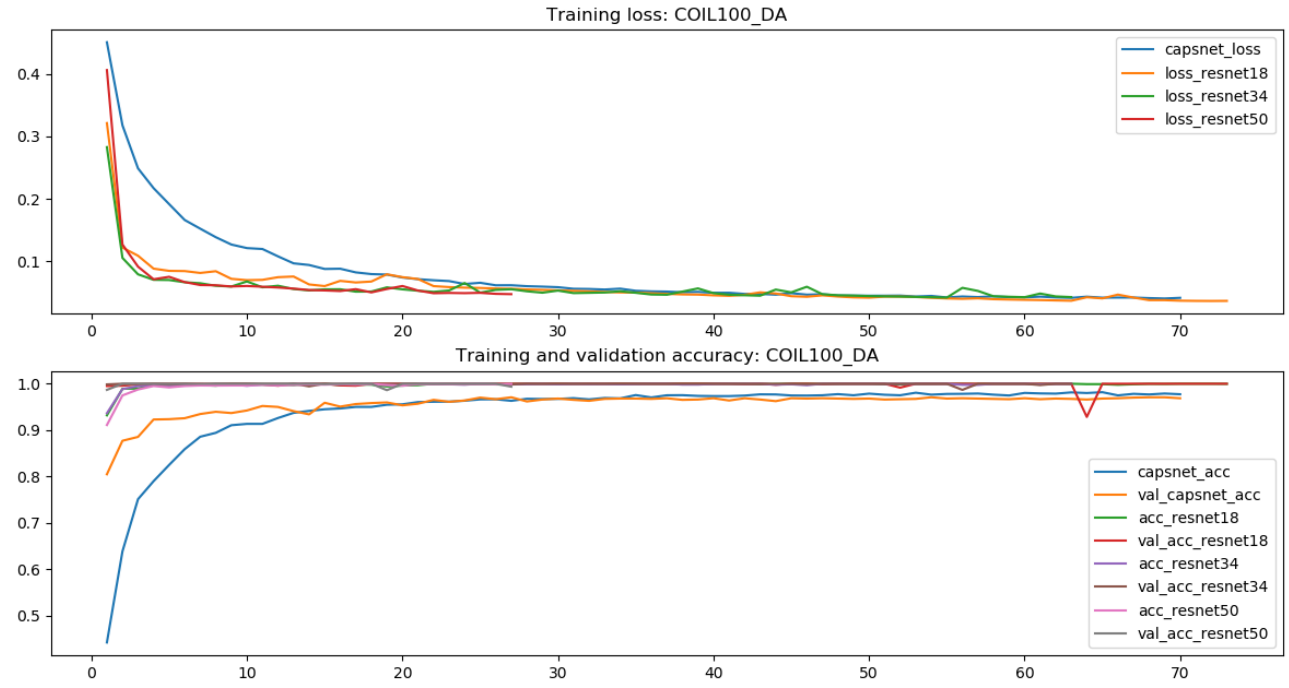

FIGURA 41. RESULTADOS DEL DATASET COIL-100 CON *DATAAUGMENTATION*

En el conjunto de datos Mnist (LeCun & Cortes, 2010), en ambos casos el mejor resultado ha sido para las redes de cápsulas como se aprecia en la Figura 42 y en la Figura 43.

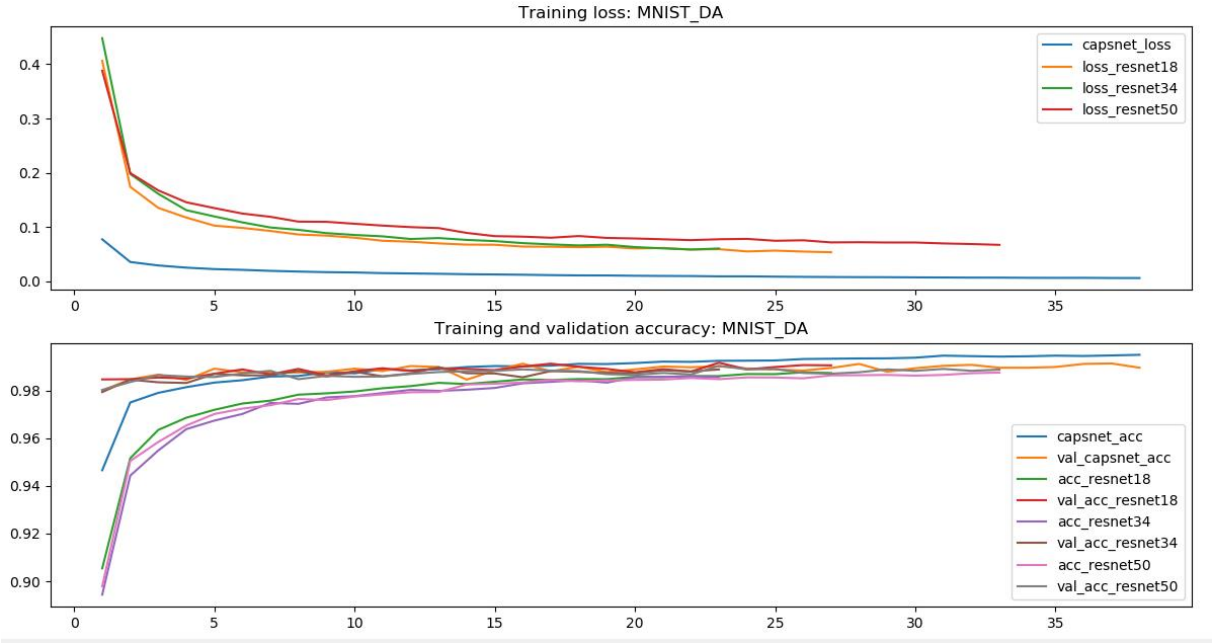

FIGURA 42. RESULTADOS DEL DATASET MNIST CON DA

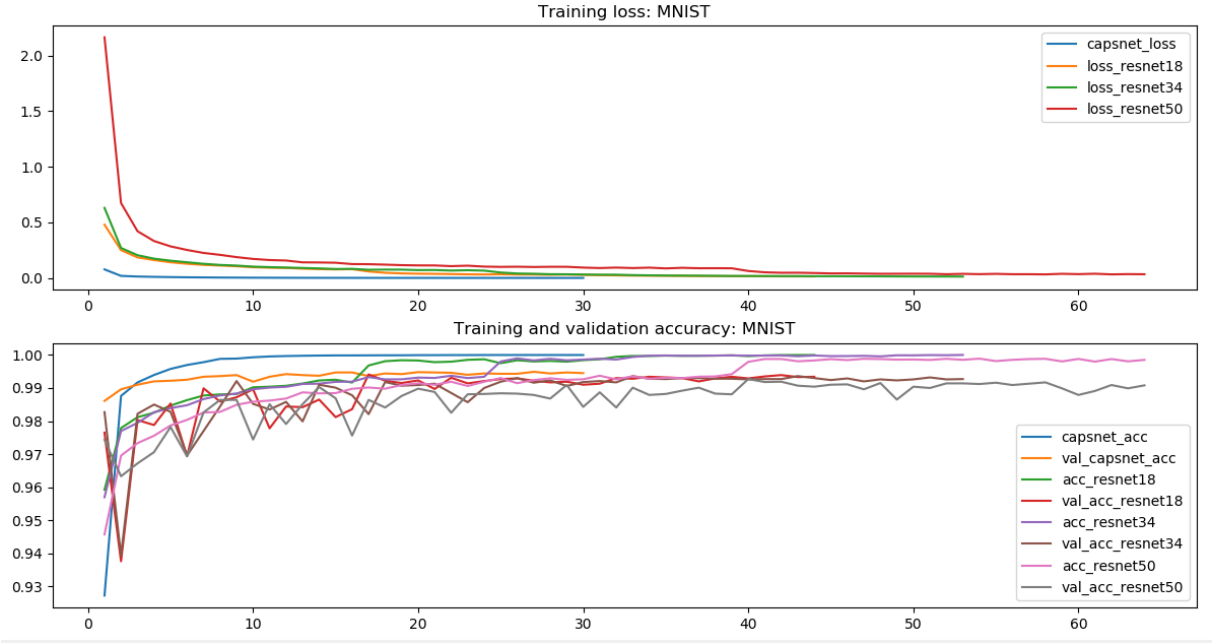

FIGURA 43. RESULTADOS DEL DATASET MNIST SIN DA

En vista de los resultados, se aprecia que el peor resultado para las redes de cápsulas ha sido para el conjunto de datos Cifar10 (Krizhevsky et al., s. f., p. 10), que es precisamente el que más imágenes tiene, y más complejas, ya que son imágenes reales, y no siempre están

aisladas. Esto nos indica que la red de cápsulas tiene que mejorar, y estar más afinada para imágenes más complejas que las de los *dataset* Mnist (LeCun & Cortes, 2010) y Fashio-Mnist.

En el conjunto más simple, Mnist (LeCun & Cortes, 2010), el modelo de redes de cápsulas ha obtenido el mejor resultado. En Fashion-Mnist (Guo, 2017, p.) ha sido el segundo mejor resultado, aunque la cantidad de tiempo invertida en el entrenamiento ha sido bastante menor que para el mejor resultado, que ha sido para Resnet50 (He et al., 2016a) en este caso.

Se puede decir, que a menor complejidad del *dataset*, mejor resultado en el modelo. Sobre la técnica *data augmentation*, donde realmente se han notado mejoras en los resultados han sido en Cifar10 (Krizhevsky et al., s. f., p. 10), que es el *dataset* más complicado.

Cabría esperar, que los modelos de redes de cápsulas tienen todavía mucho recorrido por delante, y se puede afirmar, que tienen buenas expectativas de futuro, ya que como se ha comentado antes, llevan poco tiempo siendo estudiadas.

5. Conclusiones y trabajo futuro.

Durante el desarrollo de este trabajo se ha investigado la capacidad que tienen las redes de cápsulas en la tarea de clasificación de imágenes, frente a las redes neuronales. Dentro de las redes neuronales, se ha elegido estas redes convolucionales, ya que desde hace años mantienen el estado del arte, con unos resultados en tareas de visión artificial muy superiores a cualquier otra técnica.

Para ello en este trabajo primeramente se ha presentado un estado del arte del uso de redes neuronales para la detección de imágenes.

Las redes convolucionales fueron descubiertas e implementadas, en el año 2012. Año tras año no han hecho sino mejorar los resultados en términos tanto de precisión como de error. Desde que se empezaron a estudiar e investigar, cada mejora ha supuesto un añadido en complejidad al modelo, que se ha traducido en diversos problemas como los citados *overfitting* y *curse of dimensionality* entre otros. Estos errores, se han ido solucionando con nuevas técnicas, que consistían, por ejemplo, en añadir más capas tipo *dropout* o *batch* normalization. Otra de las formas de ir mejorando los resultados de las

redes convolucionales fue ir añadiendo más capas, con más filtros, de manera que se seguía añadiendo complejidad y también una necesidad creciente de mejores recursos informáticos a la hora de entrenar estar redes. Una de las técnicas que se usan para reducir la complejidad de estos modelos es el uso de capas *max pooling*, consiguiendo así modelos más sencillos y livianos, pero perdiendo información muy importante (como, por ejemplo, la relativa a la posición del objeto).

Desde hace menos de tres años, existe un nuevo modelo, que en la teoría nos muestra que pueden solucionar todos estos problemas que presentan las redes convolucionales, que son las redes de cápsulas. Las redes de cápsulas, guardan toda la información relativa a la pose del objeto en un vector, de manera que toda esta información queda guardada y no se pierde. También, usan un *decoder*, de manera que, para la fase de entrenamiento, ellas mismas se generan unas imágenes distorsionadas, con lo que en un principio no serían necesarias tantas muestras para entrenar como las que pudiera necesitar un modelo basado en redes convolucionales.

Después del estudio del estado del arte se decidió realizar una comparativa entre las redes convolucionales y las redes de cápsulas a partir de cuatro *datasets* con características distintas.

Para realizar dicho análisis se optó por seleccionar como red convolucional el modelo ResNet (He et al., 2016a). Este modelo en el año 2015 consiguió los mejores resultados clasificando imágenes en el conocido reto ILSVRC (Russakovsky et al., 2015). Este modelo consistía en tenía una profundidad de 152 capas. En este trabajo, por razones de alcance de hardware, se han seleccionado tres modelos con distintas profundidades, 18, 34 y 50 capas convolucionales. Y para las redes de cápsulas se ha elegido el modelo basado en artículo Dynamic rounting between Capsules (Sabour et al., 2017) formado por un modelo con una primera fase, llamada encoder, formado por capa convolucional, más una capa PrimaryCap, más una última DigitCap y una segunda fase *Decoder*, formado por tres capas *fully connected*.

Con el fin de comprobar la efectividad de las redes de cápsulas, se añadió al análisis la técnica *data augmentation (DA)*, que consiste en añadir perturbaciones a los datos originales para aumentar la cantidad de datos disponibles. Lo que se pretendía analizar en este caso es si las redes de cápsulas eran igual de efectivas con los *datasets* que no contaban con DA.

Una vez realizada la selección de *datasets* y la implementación de algoritmos se realizó la comparativa de resultados. El primer resultado visible fue la drástica reducción del número de parámetros que obtuvieron las redes de cápsulas. En tiempo (duración) de entrenamientos no hubo mejoras por parte de las redes de cápsulas. Si bien, sí que se pudo ver que los modelos de ResNet (He et al., 2016a), son más complejos ya que el tiempo de entrenamiento aumenta

y, en general, los resultados mejoran. Y si se mira la precisión en la tabla de validación, se ve que las redes de cápsulas han obtenido mejores resultados en los *datasets* Mnist (LeCun & Cortes, 2010) y COIL-100 (Nene et al., 1996), no siendo así en Fashion-Mnist (Guo, 2017, p.) y Cifar10 (Krizhevsky et al., s. f., p. 10). Con respecto a si se mejora o no los resultados con DA, en validación sí que se han mejorado en general, pero aquí tampoco destacan los resultados obtenidos sin DA en las redes de cápsulas y dependiendo del modelo han mejorado o empeorado usando la técnica.

Se ha visto que con COIL-100 (Nene et al., 1996)el único modelo que ha funcionado ha sido el de las redes de cápsulas. En los modelos ResNet (He et al., 2016a) se ha obtenido *overfitting* (mejores valores en validación que en entrenamiento). Esto puede ser debido al pequeño tamaño del conjunto, por lo que las redes convolucionales se lo han *aprendido de memoria*, sin llegar a converger, ya que los resultados de validación han sido mejores que los de entrenamiento. Así, COIL-100 (Nene et al., 1996)es el conjunto que menos datos tiene. Está compuesto por 100 clases (número muy superior a las 10 clases de los otros conjuntos), y las imágenes (72 por cada una) son rotaciones de una misma. Es decir, es un conjunto para el que las redes de cápsulas, según la teoría que nos dice que necesitan de menos datos para entrenar, guardan toda la información relativa a la posición, y admiten imágenes más complejas, funcionarían mejor. Y en este caso, así ha sido. Hay que destacar, que en el estudio de clasificación de logotipos de coches (Chen et al., 2018), donde el *dataset* era similar también superaron las cápsulas de redes a las redes convolucionales. Esto puede ser debido a que las imágenes, son rotaciones, o imágenes con mucho ruido de la original, es decir, es un conjunto muy parecido al de COIL-100 (Nene et al., 1996), donde ambos, no tienen nada que ver con el conjunto Cifar10 (Krizhevsky et al., s. f., p. 10), donde las imágenes son reales, son muy distintas entre sí, y además puede haber muchas más variaciones además de la rotación y el ruido.

Así, se concluye, que en conjuntos donde se han clasificado objetos, y las muestras han sido rotaciones, o bien esos mismos objetos, pero con ruido, las redes de cápsulas han resultado ser mejores. Sin embargo, en conjuntos más reales, donde las imágenes no están tan aisladas, las redes convolucionales han obtenido los mejores resultados. A partir de este resultado en estos dos conjuntos, se podría plantear, como ampliación de este estudio, realizar otras comparativas entre este tipo de conjuntos.

Así, se tiene que los resultados no han sido tan prometedores como los esperados, ya que el desempeño en general ha sido mejor el de las redes convolucionales tipo ResNet (He et al., 2016a), que el de las redes de cápsulas, que han sido superiores en un conjunto de datos específico.

También, se ha podido comprobar que a más datos no implica mejores resultados, demostrando que las redes de cápsulas no necesitan tantos datos como las redes convolucionales.

Se concluye, que las redes de cápsulas están todavía en una fase inicial de su estudio. Si bien es verdad que han obtenido muy buenos resultados en los conjuntos de datos Mnist (LeCun & Cortes, 2010) y COIL-100 (Nene et al., 1996), pero no en los conjuntos como Cifar10 (Krizhevsky et al., s. f., p. 10)o Fashion-Mnist (Guo, 2017, p.).

No obstante, a pesar de no haber superado los resultados de las CNN tipo ResNet (He et al., 2016a) aquí probadas, los resultados han sido buenos, ya que la diferencia ha estado entre 5% y 10%, que a pesar de ser mucha diferencia, ha sido una red que ha podido converger y obtener resultados superiores al 90% en casi todos los casos.

Como trabajo a futuro, se propone para aumentar la precisión cambiar el número de características a detectar en la primera capa convolucional. También se puede experimentar añadiendo más capas de cápsulas (PrimaryCaps) para ir afinando también en la búsqueda y combinación de nuevas características.

La representación de imágenes en redes de cápsulas (la fase del *decoder*) también puede abrir nuevas vías para el desarrollo en el campo de la clasificación de imágenes, ya que ofrecen mejores y formas más sencillas de visualizar las características concretas a detectar.

Pero como se ha visto, las redes de cápsulas son más complejas de implementar que las redes convolucionales, y todavía no se implementan tan ampliamente como si lo hacen las CNN. Sin embargo, existen trabajos recientes en investigación, algunos aquí vistos, que demuestran la viabilidad del trabajo hecho por los investigadores de las redes de cápsulas. Y esto hace pensar que tengan un futuro esperanzador en el campo de las tareas de visión por ordenador.

Referencias

Basaveswara, S. K. (2019). *CNN Architectures, a Deep-dive*. https://towardsdatascience.com/cnn-architectures-a-deep-dive-a99441d18049

Calvo, Diego. (2017, julio 20). Red Neuronal Convolucional [Científica]. *Diego Calvo*. http://www.diegocalvo.es/red-neuronal-convolucional/

Chen, R., Jalal, M. A., Mihaylova, L., & Moore, R. K. (2018). Learning capsules for vehicle logo recognition. *2018 21st International Conference on Information Fusion (FUSION)*, 565–572.

Dadashzadeh, A., Targhi, A. T., Tahmasbi, M., & Mirmehdi, M. (2019). HGR-Net: A fusion network for hand gesture segmentation and recognition. *IET Computer Vision*, *13*(8), 700–707.

Deng, F., Pu, S., Chen, X., Shi, Y., Yuan, T., & Pu, S. (2018). Hyperspectral image classification with capsule network using limited training samples. *Sensors*, *18*(9), 3153.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

Deshpande, A. (2019). The 9 Deep Learning Papers You Need To Know About [Engineering]. *Adit Deshpande*. https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193–202.

Geron, A. (2018, febrero 6). Introducing capsule networks [Science]. *O'Really*. https://www.oreilly.com/ideas/introducing-capsule-networks

Guo, X. (2017). *CapsNet-Fashion-MNIST*. GitHub. https://github.com/XifengGuo/CapsNet-Fashion-MNIST

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. *European conference on computer vision*, 630–645.

Hinton, G. (2014). *What's wrong with convolutional nets?* https://techtv.mit.edu/collections/bcs/videos/30698-what-s-wrong-with-convolutional-nets.

Hinton, G. E., Krizhevsky, A., & Wang, S. D. (2011). Transforming auto-encoders. *International Conference on Artificial Neural Networks*, 44–51.

Hinton, G. E., Sabour, S., & Frosst, N. (2018). *Matrix capsules with EM routing*.

Hitawala, S. (2018). Evaluating ResNeXt Model Architecture for Image Classification. *arXiv preprint arXiv:1805.08700*.

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106–154.

Kim, Y., Wang, P., Zhu, Y., & Mihaylova, L. (2018). A capsule network for traffic speed prediction in complex road networks. *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 1–6.

Kotikalapudi, R. (2017). *Keras-resnet*. GitHub. https://github.com/raghakot/keras-resnet

Krizhevsky, A., Hinton, G., & others. (2009). *Learning multiple layers of features from tiny images*. Citeseer.

Krizhevsky, A., Nair, V., & Hinton, G. (s. f.). *CIFAR-10 (Canadian Institute for Advanced Research)*. http://www.cs.toronto.edu/ kriz/cifar.html

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., & others. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., & others. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

LeCun, Y., & Cortes, C. (2010). *MNIST handwritten digit database*. http://yann.lecun.com/exdb/mnist/

Nene, S. A., Nayar, S. K., Murase, H., & others. (1996). *Columbia object image library (coil-20)*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Sabour, S., Frosst, N., & Hinton, G. E. (2017). Dynamic routing between capsules. *Advances in neural information processing systems*, 3856–3866.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, *abs/1409.1556*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. *Thirty-First AAAI Conference on Artificial Intelligence*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Tsang, S.-H. (2018a). *Trimps-Souchen- Winner in ILSVRC 2016 (Image Classification)*. https://towardsdatascience.com/review-trimps-soushen-winner-in-ilsvrc-2016-image-classification-dfbc423111dd

Tsang, S.-H. (2018b, diciembre 12). Trimps-Souchen- Winner in ILSVRC 2016 (Image Classification) [Science & technology]. *Towards Data Science*. https://towardsdatascience.com/review-trimps-soushen-winner-in-ilsvrc-2016-image-classification-dfbc423111dd

Tsang, S.-H. (2019, mayo 8). Review: SENet—Squeeze-and-Excitation Network, Winner of ILSVRC 2017 (Image Classification) [Science & technology]. *Towards Data Science*. https://towardsdatascience.com/review-senet-squeeze-and-excitation-network-winner-of-ilsvrc-2017-image-classification-a887b98b2883

University of Amsterdan. (2016). *European Conference on Computer Vision*. http://www.eccv2016.org/

Wu, Z., Shen, C., & Van Den Hengel, A. (2019). Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, *90*, 119–133.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Zagoruyko, S., & Komodakis, N. (2016). *Wide Residual Networks*. http://arxiv.org/abs/1605.07146

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *European conference on computer vision*, 818–833.

Zhang, K., Sun, M., Han, T. X., Yuan, X., Guo, L., & Liu, T. (2017). Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, *28*(6), 1303–1314.

Zheng, Y., Yang, C., & Merkulov, A. (2018). *Breast cancer screening using convolutional neural network and follow-up digital mammography*. 4. https://doi.org/10.1117/12.2304564

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Anexo. Artículo Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes

Análisis comparativo de arquitecturas de redes neuronales para la clasificación de imágenes.

Lara Moreno Díaz-Alejo

Universidad Internacional de la Rioja, Logroño (España)

Febrero 2020

Resumen

Dentro del campo de la visión artificial, la tarea de clasificación de imágenes ha estado dominada desde hace algo más de una década, por las redes neuronales convolucionales. Estas redes, a pesar de este dominio, sufren de varios inconvenientes. Para tratar de solventar estos problemas, y haciendo un estudio de sus inconvenientes, recientemente se ha desarrollado un nuevo algoritmo de redes neuronales llamado redes de cápsulas basado en vectores y en un nuevo algoritmo de entrenamiento llamado algoritmo de enrutamiento dinámico. En este estudio hemos probado el rendimiento de esta nueva arquitectura, y la hemos comparado con una arquitectura convolucional basada en bloques residuales. Para contrastar los resultados hemos usado cuatro conjuntos de datos de distintas complejidades. A pesar de la buena perspectiva teórica que nos ofrece este nuevo modelo en la teoría, los resultados obtenidos todavía no son lo esperados en cuanto manejamos conjuntos de datos con cierta complejidad.

Palabras Clave

Redes de Cápsulas,
Redes Neuronales Convolucionales,
Bloques Residuales

I. Introducción

Introducción. La tarea de clasificación de imágenes es una de las áreas dentro del campo de visión artificial que más se ha estudiado en la última década debido al gran interés que ha generado tanto en la comunidad en general, como en la científica. Sus numerosos y llamativos campos de aplicación, como reconocimiento facial con cámaras o móviles, reconocimiento de huellas digitales y coches autónomos entre otros, evidencian además del interés, la necesidad de seguir estudiando y mejorando estos algoritmos para hacerlos más rápidos y fiables. Infinidad de estudios en los últimos años, han demostrado que las redes convolucionales son actualmente la mejor herramienta para realizar clasificación de imágenes.

La primera referencia a estas redes fue LeNet-5 [1] y data de 1998. Previamente, en 1980 ya existía algún intento (con otro nombre y sin éxito). Este algoritmo fue usado en algunos bancos para reconocimiento de dígitos. Fue toda una novedad. A partir de este año, estas redes fueron ampliamente estudiadas llegando mantener el estado del arte durante más de una década.

A pesar del reconocimiento y los grandes éxitos cosechados, las redes convolucionales tienen dos grandes debilidades, ya que por un lado no tienen en cuenta la posición de los objetos que hay en las imágenes, ni las relaciones que pueda haber entre ellos, y no tienen invarianza rotacional. También, cuando las redes son muy profundas y complejas, pueden llegar a un punto en el que ya no aprendan más, apareciendo los problemas de *vanishing gradient*, *exploding gradient* y *curse of dimensionality*.

Estudiando los problemas de las redes convolucionales, Geoffrey Hinton publicó la teoría de redes de cápsulas [2], cuya principal diferencia con las redes convolucionales es que usa vectores para guardar toda la información relativa a la posición, y las relaciones entre los distintos objetos de la imagen, evitando así perder información muy importante para luego poder realizar una buena clasificación.

Debido al potencial que nos presentan las redes de cápsulas, este estudio tiene como objetivo ver el rendimiento de estas redes en la tarea de clasificación de imágenes. Para ello, se ha hecho un estudio en profundidad del estado del arte de esta tarea centrándonos en las redes convolucionales por sus probados buenos resultados. Finalmente, se ha decidido compararlas con las redes convolucionales basadas en bloques residuales, ya que éstas últimas tienen el prestigio y un probado éxito en el ámbito de la clasificación de imágenes. Para ello, se ha utilizado la arquitectura ResNet [3], con una red que tiene diferentes profundidades. También se han estudiado y revisado los *datasets* existentes, seleccionando finalmente cuatro, Mnist [4], Fashion-Mnist [5], Cifar10 [6] y Coil100 [7]. Son de distintas complejidades y distintos tamaños para poder realizar un buen análisis comparativo.

Sin embargo, los resultados obtenidos no han sido del todo los esperados. En Mnist [4] la red de cápsulas sí que ha desempeñado una buena clasificación, incluso con el *dataset* Coil100 [7] también han demostrado ser superiores, no siendo así ni en Cifar10 [6], ni en Fashion-mnist, pudiéndose considerar estos dos últimos más complejos que los anteriores.

II. Estado del Arte

En este capítulo se da una visión histórica y teórica de las arquitecturas basadas en redes neuronales convolucionales y en redes de cápsulas. Se hará una pequeña explicación teórica, para luego recorrer a lo largo del tiempo el desarrollo y evolución que han tenido las redes neuronales convolucionales y de cápsulas. También se hará especial hincapié en cómo los problemas e inconvenientes que han sufrido las redes convolucionales han favorecido la búsqueda a alternativas a estas, dando lugar a las redes de cápsulas, por ejemplo.

Redes Convolucionales.

Las redes convolucionales [8] son algoritmos que toman imágenes como entrada y van detectando una serie de características de manera

que dependiendo de estas características que hayan detectado, pueden distinguir unas imágenes de otras, y, por ejemplo, clasificarlas. A su vez, la tarea de clasificación de imágenes, la podemos definir como la capacidad de una vez detectada una imagen saber a qué grupo pertenece. Para ello, los humanos de manera inconsciente van seleccionando características (por ejemplo, número de ruedas, número de patas en animales) y así van reconociendo patrones, siendo luego capaces de generalizar en base al conocimiento que se tiene y poder decidir qué objeto están viendo (coche, moto, o gato).

Esta tarea de clasificación, que para los humanos puede parecer sencilla y automática, para los algoritmos es mucho más compleja, ya que lo único que les llega es una matriz de números, y a partir de estos números, el algoritmo tiene que buscar los patrones básicos que le permitan reconocer de qué objeto se trata. Así, primero tendrá que detectar bordes, líneas y curvas, por ejemplo, para luego poder construir características más complejas como círculos o cuadrados, y finalmente poder detectar, por ejemplo, una rueda.

El funcionamiento básico de una red convolucional es el anteriormente descrito. Las capas convolucionales de las redes van buscando y encontrando características muy simples como líneas o curvas, para luego, a través de las capas superiores ir formando objetos más complejos como un círculo o un cuadrado, hasta poder construir objetos más abstractos y complejos, como una rueda y finalmente, por ejemplo, un coche.

Las capas convolucionales usan filtros, o kernels. Estas capas realizan la operación de convolución, que consiste en el producto escalar del filtro por un tamaño igual en la imagen, y su suma es el valor de la nueva capa. El filtro recorre toda la imagen de izquierda a derecha y de arriba abajo calculando todos los valores de salida de la capa convolucionada. Y por cada filtro, se obtiene una característica (o *feature map*).

Una vez aplicada una capa de convolución, se aplica la función de no linealidad ReLU. Después, para reducir el tamaño de la capa y que no se haga inmanejable a través de la aplicación de los sucesivos kernels, se usa la técnica de *max-pooling*. Esta técnica consiste en dividir la imagen en sub-matrices, y coger el máximo valor de cada una de ellas. Al aplicar esta operación, sólo las neuronas más activas (máximos valores) se están pasando a la siguiente capa, con la consiguiente pérdida de información. Es decir, con esta operación pierden información que pudiera ser muy importante para variaciones como modificaciones del fondo o rotaciones. Es decir, no proporcionan un aprendizaje robusto frente a distintas escalas o rotaciones. Y una de las maneras de evitar esta pérdida, es aumentar el tamaño del dataset, como veremos más adelante.

Finalmente se llega a la última capa que es la capa clasificadora llamada *Fully connected layer (FC)*. A esta capa llegan la salida de las capas convolucionales que representan características de alto nivel de la imagen (por ejemplo, cuatro ruedas). La capa completamente conectada es una forma sencilla de aprender combinaciones no lineales de estas características. De hecho, aprende una función no lineal que toma un volumen de entrada (generado por las capas de convolución y ReLU) y lo convierte en un vector N dimensional (siendo N el número de clases a clasificar). Y cada número en este vector representa la probabilidad de una determinada clase. La capa conectada lo que hace es observar el resultado de la capa anterior (los mapas de características de alto nivel) y determina qué características se relacionan más con cada clase en particular. Si el algoritmo predice por ejemplo que será un coche, tendrá valores altos en los mapas de características que representen cuatro ruedas.

Podemos resumir que las CNN son combinaciones de capas de convolución y de pooling que extraen las características principales de cada capa, con una última capa clasificadora llamada *fully connected*, encargada de realizar las predicciones. La capa de *pooling*, además de reducir el coste computacional, controla el sobreajuste (overfitting). Esta reducción en la dimensión se traduce en pérdida de información y cuando estas redes se enfrentan a datos complejos, donde las jerarquías, el orden y la posición de las características son muy relevantes, cometen fallos, y llegan a ocasionar falsos positivos y falsos negativos debido a esa falta de información.

También hay que mencionar que cuando las redes convolucionales son muy profundas y complejas dejan de aprender apareciendo tres principales problemas: la degradación/explosión del gradiente (*vanishing/exploding gradient*) y la maldición de la dimensionalidad (*curse of dimensionality*). El gradiente es el error que calcula el algoritmo de *backpropagation* durante el entrenamiento de la red, y se utiliza para actualizar los pesos en la red e ir mejorando las predicciones. Si este error se va acumulando, puede dar lugar a valores muy grandes haciendo que la red se vuelva inestable y deje de aprender. Se produce entonces la explosión del gradiente, o en inglés *exploding gradient*. Por el contrario, si este valor se va haciendo cada vez más pequeño, llega un momento en que, si se acerca mucho a 0, la red deja de aprender, y se produce lo que se llama *vanishing gradient* (el gradiente desaparece).

La maldición de la dimensionalidad (*curse of dimensionality*) se produce cuando hay demasiadas características para tener en cuenta en una red neuronal, desembocando en un tiempo de computación muy alto, y en entradas que al final son irrelevantes o redundantes, y que hacen que la arquitectura no aprenda.

Resumiendo, las redes convolucionales tienen un gran éxito, pero tienen ciertos inconvenientes que las pueden hacer poco eficientes en condiciones no óptimas, teniendo que buscar cada vez soluciones más complejas y costosas computacionalmente.

En 2011 Hinton presentó por primera vez el concepto de "Capsule networks" en el artículo "Transforming auto-encoders" [2], donde explica toda la teoría de estas *redes de cápsulas*, pero no fue hasta finales de 2017, cuando por fin presentó un método para entrenarlas, con el que consigue mejorar algunos de los resultados obtenidos por las redes convolucionales. Sobre la manera de entrenar las redes de cápsulas se han presentado dos artículos "Matrix capsules with EM" [9] y en "Dynamic route between capsules" [10].

En la teoría de las redes de cápsulas para realizar el reconocimiento de objetos se guardan las relaciones jerárquicas de las posiciones de los objetos de la imagen y lo representan de manera numérica en un vector de pose (posición). Cuando estas relaciones se construyen en una representación interna de los datos, es cuando se facilita al modelo la labor de aprender y reconocer lo que hay en la imagen.

Es decir, que las características que están relacionadas entre sí debido a su orientación y situación espacial se agrupan en estas cápsulas. Estas cápsulas realizan complicadas operaciones internas en sus entradas, y luego encapsulan el resultado en un vector que contiene toda la información de las características para la salida de la capa, de manera que la información no se pierde. Recordar, que en las redes convolucionales cada neurona tiene como salida un único escalar, resultado de aplicar los filtros a las distintas capas. Luego, en la capa de *max-pooling* se va seleccionando el número más alto de cada región, de manera que se conseguiría la invariabilidad de la capa. Pero estas operaciones, en la práctica, no terminan de funcionar de manera adecuada, porque al aplicar la capa de *max-pooling* se está perdiendo información que puede ser muy importante, ya que no codifica, ni guarda las relaciones espaciales entre las características. Y un movimiento de rotación, por ejemplo, podría dar lugar a un falso negativo. Pero las cápsulas de red, al guardar la información relativa a la posición de los objetos en un vector pueden reconocer el objeto al tener toda esa información guardada.

Estas cápsulas codifican en un vector la probabilidad de detectar una característica en la longitud del vector. El estado de la característica (dónde y cómo está colocada) es codificado como la dirección del vector. De manera que cuando una de las características cambia su posición, la probabilidad y el estado seguirán siendo los

mismos, mientras que la dirección del vector cambiará.

Si nos imaginamos un coche en 3D y lo movemos por la imagen, la dirección del vector irá cambiando (el vector rotará), detectando el cambio en el coche, pero la probabilidad del vector y su longitud seguirán siendo fijos por lo que la cápsula seguirá detectando un coche. A esto lo llama Hinton una actividad con equivalencia: cuando un objeto cambia su estado a cualquier otra forma posible y la probabilidad de detectarlo no cambia, es decir, cambia la dirección del vector. Y esto es lo que falta en las redes convolucionales.

A continuación, se presentan varios trabajos que han destacado estos últimos años por ser ganadores del reto anual en tareas de visión artificial ILSVRC, *ImageNet Large Scale Visual Recognition* [10].

En el año 2012 ganó por primera vez AlexNet [11], una primera red neuronal que utilizó una arquitectura basada en capas convolucionales. El gran éxito de esta red, además de la gran mejora en la precisión fue una bajada importante en la tase del error. Estaba compuesta por 5 capas convolucionales y 3 capas *fully connected*, con 60 millones de parámetros y 650000 neuronas.

En 2014, apareció VGGNet [12]. Esta arquitectura reforzó el razonamiento intuitivo de que para tener una red que logre una representación jerárquica suficientemente buena, tiene que tener una arquitectura de capas profundas. También destacó por su simplicidad. Formada por 19 capas convolucionales (seguidas cada una de la función ReLU, y *max pooling* correspondientes) consiguió una arquitectura profunda y simple. Mejorando la tasa de error del momento.

En 2015 destacó GoogLeNet [13], cuyo enfoque era el contrario que VGGNet [12]. Fue la primera red que no apiló capas convolucionales una encima de otra. En esta red aparece el concepto de módulo *Inception*, gracias al cual se paralelizan los procesos. Estos módulos *Inception*, permiten realizar las operaciones de las capas convolucionales en paralelo y con tamaños diferentes de kernel, distintos números de convoluciones y de capas de *max pooling*, de manera que los tamaños de los volúmenes no resultan excesivos y el número de parámetros se reduce drásticamente. Otra diferencia es que elimina la capa *fully connected*, aplicando en su lugar, el método *global average media.*

Siguiendo el camino de los bloques, al año siguiente Microsoft presentó una red convolucional basada en bloques residuales, ResNet [3]. Con los bloques residuales se consiguió que las 152 capas que componían la red, tuvieran menos complejidad al entrenar que VGGNet [12]. De nuevo, se batieron records de clasificación, localización y detección de objetos. La idea base de ResNet [3], es porqué apilar bloques y bloques, si está demostrado que a partir de cierta profundidad la red no siempre mejora, y aparecen problemas como *vanishing gradient* o *curse of dimensionality*. La idea básica de los bloques residuales es saltar unas cuantas capas, introduciendo una conexión residual (con una capa identidad). Y sus autores afirman que el proceso de aprendizaje mejora, ya que esta capa va directamente a la siguiente.

Se puede apreciar que las redes convolucionales son cada vez más complejas. Se buscan combinaciones nuevas, nuevos bloques o nuevas operaciones sobre los bloques ya conocidos buscando nuevos logros y mejorar el estado del arte. Por eso, el nuevo enfoque que da la teoría de redes de cápsulas, que enfatiza en no perder información desde un principio para poder aprovecharla y conseguir mejorar los resultados, resulta tan interesante de estudiar y probar.

Redes de cápsulas.

Hinton, da un nuevo enfoque sobre estas redes y la manera en que aprenden los mapas de características. Se van a analizar los trabajos presentados por su equipo, explicando la teoría de cápsulas, y otros que poco a poco van viendo la luz aportando mejoras sobre los ya existentes.

El artículo *Transforming Auto-encoders* [2] es el primer artículo que define y explica las redes de cápsulas. Se presenta como una alternativa que mejora las soluciones existentes, tanto basadas en redes neuronales como los algoritmos diseñados a mano y específicos como SIFT [14]. La principal ventaja que presentan las redes de cápsulas según este estudio es que guardan toda la información relativa a la pose, es decir, posición, tamaño, orientación o escalado son algunas, en un vector, de manera que estas propiedades que acabamos de citar, serían la dirección del vector. Y su valor, sería la probabilidad de existencia de la característica en cuestión, guardándose en la longitud del vector. De esta manera, cada cápsula (grupo de neuronas) tendría su vector con toda la información relativa a la característica correspondiente y la probabilidad de existencia de la característica. Es decir, cada cápsula aprenderá a reconocer una característica en una región limitada de la imagen, generando su vector de salida donde guardará toda la información relativa a la pose, y la probabilidad de existencia de la propia característica. De manera que, si el objeto cambia algo con respecto a su pose, por ejemplo, es girado, hay un cambio en la luz, es ampliado, movido o rotado, la dirección del vector salida variará, indicando tal modificación, pero la probabilidad de existencia de dicha característica no lo hará, conservando la propiedad de invariabilidad. Una ventaja de trabajar con este tipo de vectores, es que así las cápsulas aprenden a reconocer el todo (de la imagen) a partir de las partes. De manera que si una cápsula aprende a reconocer la pose de su objeto (de una característica), entonces será posible determinar si las características que están representadas por otras cápsulas tienen una relación espacial correcta para que se active una cápsula de nivel superior.

Ahora quedaría saber cómo generar la primera capa de cápsulas. Y esto se hace por medio del diseño de un *transforming auto-encoder*. La idea subyacente, es que por cada entrada se genera una salida distorsionada, sumándole un incremento a cada coordenada y dejando su probabilidad p. De manera que la salida es otra imagen cambiada con su correspondiente probabilidad. Cada entrada al *auto-encoder* generará varias imágenes de salida, obteniendo esta primera capa de cápsulas. Estas redes, al generar sus propias imágenes transformadas, necesitarán de muchos menos ejemplos para entrenar.

Una vez que se han definido las cápsulas y cómo funcionan falta ver cómo se entrenan. Pero no fue hasta el año 2017 cuando aparece el artículo *Dynamic Routing Between Capsules* [15] donde se describe e implementa un método para entrenar esta arquitectura. El estudio asume que la información de la imagen está guardada de manera que se dispone de toda la información relativa a la pose, de manera que esta información se conserva y no se pierde.

En el algoritmo de enrutamiento dinámico, cada cápsula tiene que elegir qué cápsula del nivel superior será su padre en el árbol. Para las capas altas del sistema, este procedimiento iterativo se resuelve por la pertenencia de las partes al conjunto entero. La actividad de las neuronas dentro de una cápsula representa algunas propiedades de la parte que esté presente en la imagen. Estas propiedades pueden ser la posición, tamaño, orientación, deformación, velocidad, albedo, tonalidad y textura entre otras, y se guardan en la dirección del vector. Y la probabilidad de existencia de esa parte, o característica de la imagen, será la longitud del vector, y será un valor entre cero y uno. Así, el vector salida de la cápsula hace posible usar un potente algoritmo de enrutamiento para asegurar que la salida de la cápsula es mandada al padre adecuado en la capa superior de cápsulas. En un principio, el vector se envía a todos los posibles padres, y por cada posibilidad la cápsula calcula un vector de predicción multiplicando el vector por una matriz de pesos. Si el vector resultante tiene una predicción suficientemente grande, existe una retroalimentación de arriba a abajo de manera que el coeficiente de acoplamiento de ese padre aumenta, y para el resto lo disminuye. De esta manera se aumenta la contribución que esa cápsula hace a ese padre, aumentando más el producto escalar dela predicción de la cápsula con la salida del padre. Este método debería ser más efectivo que el enrutamiento por

agrupación máxima (*max pooling*) donde se ignoran todos los detectores de características excepto los más activos, con la consecuente pérdida de información.

Una notable diferencia entre las redes de cápsulas y las redes normales es que la activación de la cápsula está basada en la comparación entre múltiples entradas de la predicción de su pose, mientras que, en una red neuronal normal, está basada en la comparación de una única entrada del vector actividad y su vector de pesos aprendido.

Una aplicación de las redes de cápsulas al mundo real, mejorando el resultado de las redes convolucionales es el artículo *A capsule network for traffic speed prediction in complex road* [16] del año 2018. En este estudio tratan los datos de tráfico como si fueran imágenes, ya que los datos de tráfico espacio-tiempo los tratan como si fuera una imagen donde estos datos se expresan en un espacio 3D donde los ejes son el espacio y el tiempo para poder asimilar los datos entre tráfico e imágenes.

El artículo da una solución con una arquitectura de redes de cápsulas para la predicción de la velocidad del tráfico en redes viales complejas. A pesar de que el conjunto de datos no es muy elevado, porque los sensores de velocidad no están por toda la red donde se analizan los datos, proponen una red neuronal de cápsulas, evitando la operación de *max pooling* y utilizando el algoritmo de enrutamiento dinámico. Los autores consiguen una mejora del 13% respecto a una red convolucional, usando datos reales del tráfico de la ciudad de Santander.

Otro estudio, *Hyperspectral image classification with capsule network using limited training samples* [17] presenta una arquitectura de dos capas de redes de cápsulas entrenada con un número de ejemplos limitado para clasificar imágenes hiperespectrales (HSI), inspirada en los modelos de aprendizaje profundos más superfeciciales. Han usado dos conjuntos de entrenamiento de imágenes HSI reales donde hay ejemplos de toda complejidad. El principal objetivo de este estudio es crear un paradigma comparable entre las redes convolucionales y las redes de cásulas, eliminando las diferencias y dejando ambas arquitecturas muy parecidas, con la diferencia de la arquitectura base. También han hecho un intento de mejorar las redes de cápsulas para conseguir los mismos resultados que han obtenido con la red convolucional. Y dado que las arquitecturas más profundas no siempre dan los mejores resultados, han diseñado un modelo más bien pequeño y poco profundo para ambas arquitecturas.

Y por último se presenta un artículo sobre la clasificación de logotipos de vehículos [18]. El reconocimiento de estos logotipos es parte esencial en los sistemas inteligentes de transporte. Esta tarea, como todas las de clasificación de imágenes, ha estado dominada en los últimos años por las redes convolucionales. Pero las redes convolucionales fallan ante los cambios de orientación, rotaciones o ruidos por citar algunas. El artículo se basa en que las redes de cápsulas guardan la información en un vector cuyos parámetros de instanciación son por ejemplo, las posiciones y las orientaciones de las imágenes y sus elementos, y en que el algoritmo empeleado para entrenar la red se realiza mediante un enrutamiento dinámico, evitando la operación de *max pooling*, con la consecuente pérdida de información que conlleva.

Así el estudio ofrece una arquitectura basada en redes de cápsulas para la clasificación de logotipos de coches, donde la precisión alcanza el 100% de precisión, superando así a las redes convolucionales.

Se ha hecho un repaso donde se pueden ver los buenos resultados de las redes convolucionales, como las deficiencias que presentan. También se ha hecho un estudio sobre las redes de cápsulas y el potencial que presentan. Las principales ventajas de las redes de cápsulas frente a las redes convolucionales son:

- No necesitan para el entrenamiento gran cantidad de datos (ni usar redes pre-entrenadas con esos datos).
- En la fase de entrenamiento no pierden información. Las redes convolucionales pierden mucha información en sus capas de pooling, que luego resuelven construyendo complejas arquitecturas para recueprar esa información perdida.
- Manejan bien las ambigüedades, debido a que guardan en un vector toda la información relativa a su estado espacial. Con las cápsulas, al mantener toda la información espacial relativa a los objetos, y las relaciones que hay entre ellos, esto deja de suceder. Y las CNN necesitan de componentes extra para poder identificar qué parte pertenece a qué objeto, mientras que las cápsulas al almacenar la jerarquía entre los objetos de la imagen y sus componentes con toda la información relativa a su posición, esto no sucede.

Pero las redes de cápsulas están todavía lejos de obtener los espectaculares resultados que si han estado obteniendo las redes convolucionales durante los últimos años. También, hay que decir que son muy costosas computacionalmente de entrenar, y que cuando hay dos objetos muy cerca, no consiguen distinguirlos. Pese a ello, presentan unas ideas muy prometedoras y pudiera ser que se esté al inicio de una nueva era con respecto al reconocimiento de imágenes. Hay que recordar que las redes convolucionales aparecieron en 1998, aunque no fuera hasta el 2012 cuando empezaron a dar unos resultados innovadores y excelentes. Y las cápsulas aparecieron en 2012, y el primer algoritmo para entrenarlas en 2017.

III. Objetivos y Metodología

El objetivo general de este estudio ha sido ver el rendimiento que tienen las redes de cápsulas. Para lograr este objetivo se ha realizado una comparativa entre un modelo de redes de cápsulas y tres modelos de una misma arquitectura de una red convolucional basada en bloques residuales con distinta profundidad. Las pruebas se han hecho sobre cuatro *datasets*, y los resultados obtenidos se han comparado.

Para alcanzar el objetivo general, ha sido necesario conseguir los siguientes objetivos específicos:

1. Revisión del estado del arte para identificar cuáles son las principales arquitecturas de redes neuronales.
2. Búsqueda y revisión de arquitecturas basadas en bloques residuales.
3. Búsqueda y revisión de arquitecturas basada en redes de cápsulas.
4. Búsqueda y estudio de las distintas herramientas, librerías y frameworks para el desarrollo de los algoritmos de las redes neuronales a seleccionar.
5. Estudio, revisión y modificación de los algoritmos seleccionados en código fuente.
6. Búsqueda y estudio de conjuntos de datos a utilizar con los algoritmos seleccionados.

La metodología que se ha llevado a cabo para lograr el objetivo de este estudio está basada en fases, y consta de las siguientes etapas:

- Fase 1. Estudio y selección de los diferentes modelos a implementar y modificar. Se realiza una detallada y profunda búsqueda bibliográfica de diferentes redes neuronales, centrándonos en las redes convolucionales. Finalmente se seleccionó la arquitectura ResNet [3] con diferentes profundidades: ResNet18, ResNet34 y ResNet50. El modelo de redes de cápsulas está basado en su estudio original [2].
- Fase 2. Búsqueda y estudio detallado de los diferentes *datasets* para la realización de las pruebas. Se ha

buscado que los conjuntos tengan diferentes complejidades y diferentes tamaños.

- Fase 3. Implementación y desarrollo de los algoritmos que han sido seleccionados. Se recogen y estudian los algoritmos para desarrollarlos y complementarlos en código Python para poder ejecutarlos sobre los distintos *data sets*.
- Fase 4. Ejecución y estudio de los resultados. Se ha ejecutado en esta etapa todos y cada uno de los modelos seleccionados con todos los conjuntos de datos. En esta fase se utilizó la técnia *data* augmentacion, multiplicando por dos las pruebas para cada conjunto de datos. Esta técnica consiste en añadir perturbaciones a los datos originales para aumentar la cantidad de datos disponibles. De cada conjunto, también, se seleccionaron datos de entrenamiento y datos de validación para ver cuál era la eficiencia y qué diferencia de tiempos había habido en cada modelo.

IV. Contribución

La principal contribución de este estudio ha sido comprobar el rendimiento de las redes de cápsulas en la tarea de clasificación de imágenes. Desde hace años el estado del arte de esta tarea ha estado dominado por las redes convolucionales, pero con serios inconvenientes cuando las imágenes no siguen la norma, es decir, están rotadas, tienen reuido, los elementos no siguen el orden pre-establecido o bien las imágenes no son de buena calidad por citar algunos. La teoría de las redes de cápsulas nos presenta ciertas ventajas frente a las redes convolucionales, como la no pérdida de información relativa a la pose de los elementos que hay en las imágenes, o la no pérdida de la información relativa a las relaciones que hay entre distintos elementos de la imagen, o bien como la reducción del tamaño de la muestra a entrenar entre algunas.

En este estudio, se ha podido comprobar que las redes de cápsulas superan a las convolucionales basadas en bloques residuales en el conjunto Mnist [4] y en el conjunto Coil100 [7], no siendo así en Cifar10 [6]. Pero no se ha podido demostrar que las redes de cápsulas superan en general a las redes convolucionales, ya que en conjuntos de datos más complejos y extensos como Cifar10 [6] o bien Fashion-Mnist [5]las redes convolucionales han conseguido los mejores resultados.

V. Resultados o Evaluación

Una vez que los cuatro modelos, CapsNet, basado en redes de cápsulas y los tres basados en redes convolucionales con bloques residuales, ResNet18, ResNet34 y ResNet50 han sido entrenados en los cuatro*data sets*, Mnist [4], Fashion-Mnist [5], Cifar10 [6] y Coil100 [7] tanto con *data augmentation* como sin ella, los resultados obtenidos se han presentado en las tablas que vienen a continuación.

En esta primera tabla, Tabla 1Tabla 1 Parámetros., vemos la diferencia de parámetros entre las redes de cápsulas, que tienen bastante menos que los correspondientes modelos basados en ResNet [3].

	Parámetros
CapsuleNetwork	8.215.568
ResNet18	11.186.186
ResNet34	21.305.482
ResNet50	23.586.570

Tabla 1 Parámetros.

A continuación, se han presentado los resultados de validación de los entrenamientos.

Acc/loss	Fashion	Mnist	Cifar10	Coil100
CapsNet	0.923/0.0639	0.9945/0.0099 sin da	0.6818/0.235	0.9688/0.0114
ResNet18	0.9248/0.2869	0.9905/0.0452*	0.8603/0.6846	Overfitting
ResNet34	0.9281/0.2863	0.9901/0.0513 *	0.8628/0.6820	Overfitting
ResNet50	0.9258/0.2862	0.9908/0.0635	0.8724/0.5932	Overfitting

Tabla 2 Resultados de validación

En la Tabla 2, tenemos que la mayoría de los resultados han sido mejores usando *data augmentation* (DA), a excepción de mnist, donde los resultados han sido muy similares (marcados con *) o incluso han sido mejor sin usarla en el caso de las cápsulas de redes.

Para el conjunto de datos Fasion-mnist todos los modelos han conseguido valores por encima del 92% de precisión, y el error ha estado en torno a 0.28, exceptuando las cápsulas que ha sido inferior. Y para todos los casos, el resultado ha sido mejor con DA. Para este *data set*, el mejor resultado ha sido para ResNet50, siguiéndole las cápsulas de redes.

Mnist ha sido el *data* set que mejor precisión ha conseguido, siendo superior al 99%, y las redes de cápsulas han sido las que mejor resultado han obtenido. Los resultados mejores se han conseguido en el modelo ResNet18 y las redes de cápsulas, donde han obtenido mejor resultado sin DA, mientras que los modelos más profundos, ResNet34 y ResNet50, han necesitado de DA (más datos) para obtener mejores resultados.

Para Cifar10 [6], estos resultados ya cambian. Las redes convolucionales han obtenido resultados sobre el 86% de precisión, frente a un 68% de las cápsulas. Y todos los mejores resultados han usado DA. Este conjunto al estar fomrmado por imágenes reales, es el más complejo, y es el que peores resultados en general ha obtenido comparado con los demás conjuntos. Y aquí se podría hacer notar que las redes convolucionales llevan una década siendo estudiadas y probadas, mientras que las redes de cápsulas apenas llevan dos años, con lo que es muy posible que a futuro tengan más recorrido.

Coil100 [7] ha sido el único conjunto que donde ha funcionado ha sido el de las redes de cápsulas. En los modelos ResNet [3] se ha obtenido overfitting (mejores valores en validación que en entrenamiento). Precisamente Coil100 [7] es el conjunto que menos datos tiene. Está compuesto por 100 clases (número muy superior a las 10 clases de los otros conjuntos), y las imágenes (72 por cada una) son rotaciones de una misma. Es decir, es un conjunto para el que las redes de cápsulas, según la teoría que nos dice que necesitan de menos datos para entrenar, guardan toda la información relativa a la posición, y admiten imágenes más complejas, funcionarían mejor. Y en este caso, así ha sido. Hay que destacar también, que el tiempo empleado en el entrenamiento ha sido muy elevado.

Este último resultado, encaja también con el resultado que obtuvieron en el estudio de los logotipos de vehículos [18].

VI. Discusión o Análisis de Resultados

En vista de los resultados, se ha visto que el peor resultado para las redes de cápsulas ha sido con el conjunto Cifar10 [6], y el mejor resultado ha sido para el conjunto Mnist [4].

La importancia de no haber superado las redes de cápsulas a las

convolucionales en Cifar10 [6], reside en que este conjunto es un conjunto de imágenes reales, donde no están aisladas, y donde realmente se puede ver si un modelo funciona o no. Mnist [4], sin embargo, es un conjunto muy sencillo de imágenes, donde sólo hay un número en cada imagen. Podríamos decir que encontrar un modelo que funcione para Mnist [4] es mucho más sencillo que para Cifar10 [6] debido a la complejidad de este último conjunto. Con Coil100 [7], donde el mejor resultado ha sido para las redes de cápsulas, es un buen ejemplo donde se cumple la teoría de las redes de cápsulas [2] y [15]: tiene un tamaño reducido de muestra, y los distintos ejemplos de cada clase, son rotaciones con distinto ángulo del mismo objeto. En este conjunto, es donde a priori se ha pensado que tendrían mejor desempeño las redes de cápsulas, y así ha sido.

VII. Conclusiones

Durante el desarrollo del presente trabajo se ha investigado la capacidad que tienen las redes de cápsulas frente a las redes neuronales en la tarea de clasificación de imágenes. Esto ha sido así, debido a que las redes convolucionales mantienen el estado del arte desde hace muchos años con unos resultados en esta tarea muy superiores a los de cualquier otra técnica.

Par ello, se ha presentado el estado del arte en el uso de redes neuronales para la clasificación de imágenes.

Las redes convolucionales se implementaron por primera vez en el año 1998. A partir de ese año sufrieron una gran evolución, gracias también a los grandes avances en computación que ha habido, los cuales han permitido obtener en tiempos razonables, cada vez mejores resultados en términos de precisión y también de error. Cada mejora, ha supuesto un incremento en la complejidad de la red, que se ha traducido en diversos inconvenientes: mayor necesidad de recursos de hardware a la hora de entrenar las redes, *overfitting* y *curse of dimensionality*, entre otros.

Desde hace menos de tres años, se ha empezado a estudiar un nuevo modelo de redes neuronales, llamado redes de cápsulas que nos muestra la teoría que pueden solucionar gran parte de los inconvenientes sufridos por las redes neuronales. Esto se debe a que guardan toda la información relativa a la posición, y las relaciones entre los distintos objetos de la imagen entre otras, en un vector. En la fase de entrenamiento, no usan el *agrupamiento*, o capas de *max pooling* usado en las redes convolucionales donde se pierde importante información. Usan un *decoder*, que genera imágenes distorsionadas a partir de la original, con lo cual, tampoco sería necesario un número tamaño de *data set* tan grande como lo sería para las redes convolucionales.

Una vez realizado el estudio del arte, se ha decidido realizar una comparativa entre las redes convolucionales y las redes de cápsulas a partir de cuatro *data sets* con características distintas.

Para llevar a cabo el análisis se seleccionó como red convolucional un modelo basado en bloques residuales, ya que este modelo, en 2015 consiguió los mejores resultados clasificando imágenes en el ILSVRC [10]. El modelo tenía 152 capas de profundidad. Para este estudio, se ha elegido el mismo modelo pero con distintos números de capas: 18, 34 y 50. Para las redes de cápsulas se ha elegido el mismo modelo del estudio original Dynamic routing between Capsules [15].

Con el fin de comprobar la efectividad de las redes de cápsulas, se ha decidido realizar dos entrenamientos, uno con *data augmentation*, y otro sin ella.

Los resultados en validación, han demostrado las redes de cápsulas superan las redes convolucionales basadas en bloques residuales con los conjuntos Mnist [4] y Coil100 [7], mientras que para Cifar10 [6] y Fashion-Mnist [5], los resultados han sido mejores en las redes convolucionales.

Se ha visto, que para el conjunto de datos el único modelo que ha conseguido converger ha sido Coil100 [7]. Esto puede ser debido al pequeño tamaño del conjunto, por lo que las redes convolucionales se lo han *aprendido de memoria*, sin llegar a converger, ya que los resultados de validación han sido mejores que los de entrenamiento. Las características de este conjunto de datos, rotaciones de un mismo objeto, junto con un tamaño de conjunto de datos reducido hacen que sea un conjunto ideal para entrenar las redes de cápsulas. También, en el artículo de clasificación de logotipos de vehículos [18] obtienen mejor resultado las redes de cápsulas, siendo el conjunto de datos similar al de Coil100 [7]. Son variaciones, con rotación y ruido añadido de una misma imagen, la cual se corresponde con el logotipo del vehículo en cuestión.

Así, se tiene que los resultados finalmente no han sido los esperados, ya que el desempeño en general de las redes de cápsulas ha sido inferior en general que el de las redes convolucionales.

También se ha comprobado que a más datos no implica mejores resultados, demostrando para las redes de cápsulas no son necesarios tantos datos como para las redes convolucionales.

Concluimos que las redes de cápsulas están en una fase muy inicial de su investigación y estudio. Si bien es cierto, que los resultados en Cifar10 [6] y en Fashion-Mnist [5] no han sido satisfactorios, en los conjuntos Coil100 [7] y Mnist [4] sí que se han obtenido muy buenos resultados, de manera que auguran un futuro prometedor

Como líneas a seguir para un trabajo futuro se propone para aumentar la precisión cambiar el número de características a detectar en la primera capa convolucional. Se puede experimentar también añadiendo más capas de cápsulas (PrimaryCaps) para ir afinando también en la búsqueda y combinación de nuevas características.

En la fase del *decoder* ofrece poder usar una visualización de manera sencilla, de forma que pueda abrir nuevas vías para el desarrollo en el campo de la clasificación de imágenes.

Pero como hemos visto, las redes de cápsulas son más complejas de implementar que las redes neuronales convolucionales, y todavía no se implementan tan ágil y ampliamente como sí lo hacen estas segundas. Sin embargo, existen trabajos recientes en investigación, algunos vistos aquí, que demuestran la viabilidad del trabajo hecho por los investigadores de las redes de cápsulas. Y esto hacen pensar que tengan un futuro esperanzador en el campo de las tareas de visión artificial.

Referencias

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, y others, «Gradient-based learning applied to document recognition», *Proc. IEEE*, vol. 86, n.º 11, pp. 2278–2324, 1998.

[2] G. E. Hinton, A. Krizhevsky, y S. D. Wang, «Transforming auto-encoders», en *International Conference on Artificial Neural Networks*, 2011, pp. 44–51.

[3] K. He, X. Zhang, S. Ren, y J. Sun, «Deep residual learning for image recognition», en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] Y. LeCun y C. Cortes, «MNIST handwritten digit database», 2010.

[5] H. Xiao, K. Rasul, y R. Vollgraf, *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017.

[6] A. Krizhevsky, V. Nair, y G. Hinton, «CIFAR-10 (Canadian Institute for Advanced Research)».

[7] S. A. Nene, S. K. Nayar, H. Murase, y others, «Columbia object image library (coil-20)», 1996.

[8] S. K. Basaveswara, *CNN Architectures, a Deep-dive*. 2019.

[9] G. E. Hinton, S. Sabour, y N. Frosst, «Matrix capsules with EM routing», 2018.

[10] O. Russakovsky *et al.*, «ImageNet Large Scale Visual

Recognition Challenge», *Int. J. Comput. Vis. IJCV*, vol. 115, n.º 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[11] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «Imagenet classification with deep convolutional neural networks», en *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] K. Simonyan y A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition», *CoRR*, vol. abs/1409.1556, 2014.

[13] C. Szegedy *et al.*, «Going deeper with convolutions», en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[14] D. G. Lowe, «Distinctive image features from scale-invariant keypoints», *Int. J. Comput. Vis.*, vol. 60, n.º 2, pp. 91–110, 2004.

[15] S. Sabour, N. Frosst, y G. E. Hinton, «Dynamic routing between capsules», en *Advances in neural information processing systems*, 2017, pp. 3856–3866.

[16] Y. Kim, P. Wang, Y. Zhu, y L. Mihaylova, «A capsule network for traffic speed prediction in complex road networks», en *2018 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, 2018, pp. 1–6.

[17] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, y S. Pu, «Hyperspectral image classification with capsule network using limited training samples», *Sensors*, vol. 18, n.º 9, p. 3153, 2018.

[18] R. Chen, M. A. Jalal, L. Mihaylova, y R. K. Moore, «Learning capsules for vehicle logo recognition», en *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 565–572.