Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TRABAJO FIN DE MÁSTER N.º TFM

Reconocimiento de caracteres en imágenes mediante el uso de CNNs

Samira Ahmed El Battioui
TUTOR: Daniel Fernando García Martínez

Febrero de 2019

Contenido

Ilustraciones

1. Introducción

En los últimos años el desarrollo e impacto que ha cosechado la utilización del paradigma Deep Learning en cuanto a la detección de objetos y anotación automática de imágenes ha propiciado el comienzo de su aplicación en el ámbito empresarial e industrial. Planteando de este modo, un cambio en las arquitecturas y tecnologías utilizadas para el desarrollo de aplicaciones de visión artificial. Y convirtiéndose en una gran tarea de investigación en la cual se fusiona Deep Learning y visión artificial. La firmeza de la apuesta se debe a la mejora de los resultados obtenidos en comparación con las tecnologías hasta ahora utilizadas, pudiendo lograr el procesamiento de altos volúmenes de imágenes con resultados capaces de superar las capacidades humanas en algunos problemas determinados.

Dentro del paradigma Deep Learning el modelo que mejores resultados ha certificado en el campo de la visión artificial en la actualidad son las Redes Neuronales Convolucionales, denominadas comúnmente CNNs (Convolutional Neural Networks). No obstante, debido a diversos factores tecnológicos ha sido relativamente hace pocos años cuando se ha comenzado a implementar estas tecnologías, ya que anteriormente no eran aplicables. Para encontrar la primera aparición en la literatura de las CNNs hay que remontarse hasta 1965 [Schmidhuber2015] y el sistema Group Method of Data Handling (GMDH), un sistema poco definido, el cual tenía tantas capas como el problema lo requiriese. Así, en 1971 [Ivakhnenko1970] cuando se describió una red GMDH profunda de 8 capas.

Estos datos dejan entrever la gran diferencia entre el marco teórico y el marco práctico que persigue a esta tecnología, siendo relativamente simple comprender las razones. Y es que, la puesta en práctica de estos modelos de manera satisfactoria requiere que se cumplan unas restricciones. En primer lugar, el uso de una gran cantidad de datos etiquetados. Con el objetivo de obtener resultados fiables y satisfactorios. Hasta hace relativamente poco tiempo, estos conjuntos de datos no existían y no eran tan fácil recolectar, mucho menos almacenarlos y manejarlos. Además, el conjunto de datos será manejado y utilizado para realizar operaciones complejas y cálculos no triviales, lo que requiere una capacidad de procesamiento disponible muy alto. Características que las máquinas existentes en los años iniciales de la tecnología no podían hacer frente. Sin embargo, no solo se necesita que la capacidad de procesamiento disponible (CPUs/GPUs) sea grande para llevar a cabo el procesamiento, sino para reducir los tiempos de cálculo hasta valores razonables para los que merezca la pena hacer uso de todos los recursos nombrados.

Así mismo, una vez atravesada la barrera de las limitaciones de recursos computacionales y de datos, el factor que ha motivado el gran desarrollo de estos algoritmos frente a otros, son los buenos resultados obtenidos en el campo de la detección de objetos y clasificación de estos, superando de manera amplia lo desarrollado hasta dicho momento. Uno de los ejemplos más significativos fue [Hinton2011] que mostraba una mejora respecto al estado

del arte en la base de datos ImageNet 2010, de este modo, en posteriores estudios en los que se ha usado redes similares, la línea de resultados ha mejorado sucesivamente.

Dentro del gran abanico de modelos y objetos a reconocer partiendo de una imagen, uno de los principales objetivos de gran número de estudios es el reconocimiento óptico de caracteres (OCR), motivado este hecho por la digitalización de gran cantidad de documentos antiguos, manuscritos, libros, materiales, etc… debido al gran número de ventajas que ofrece mantener los archivos digitales frente a mantener otro tipo de soportes.

Por su parte, el sector industrial también requiere aprovechar los avances tecnológicos en vista de poder agilizar algunas tareas que no requieren ninguna cualificación por parte de los operarios, sino un elevado número de horas infrautilizadas, realizando labores de supervisión de imágenes para detectar posibles defectos o anomalías.

Por lo tanto, el presente trabajo, trata de definir una red neuronal convolucional que realice reconocimiento de caracteres partiendo de imágenes de escenas reales en las que las condiciones de iluminación, nitidez, orientación, fondo, etc… no son óptimas debido al entorno físico de su procedencia.

De manera general, los problemas que se trata de resolver mediante el uso de redes neuronales varían en gran medida dependiendo del contenido de las imágenes y sobre todo del resultado que se pretende alcanzar, diferenciando entre la tarea detección y la tarea de reconocimiento y clasificación de estas.

Al tratarse de una tarea de reconocimiento, las imágenes deben de cumplir unas restricciones. Si bien, se ha comentado que las imágenes de partida se caracterizan por tener aspectos como una mala iluminación, fondo borroso, etc, además hay que añadir, que se trata de imágenes que contienen una secuencia de dígitos, convirtiéndose en un problema al que aún no se le ha encontrado una única solución claramente definida en la literatura, por lo tanto, no se puede partir de las imágenes originales proporcionadas.

Así, en vista de la situación, se realiza de manera previa un preprocesamiento de estas, dando como resultado un conjunto de imágenes en las que aparece un solo carácter, con las características descritas. De este modo, se realizará el diseño de una red neuronal que de solución a la tarea de reconocimiento de cada carácter impreso en la imagen.

2. Objetivos

El objetivo definido para la realización del presente proyecto se basa en la utilización de Redes Neuronales Convolucionales para el Reconocimeinto Óptico de Caracteres, comenzando por la realización de un estudio de las tecnologías a utilizar debido a que son nuevas. De este modo, el proyecto se puede dividir en diferentes fases, cuyo desarrollo permite el avance hacia la siguiente fase.

El comienzo del proyecto se centra en la introducción y estudio del funcionamiento de las Redes Neuronales Convolucionales y sus aplicaciones en el campo de visión artificial, siendo el objetivo conocer las ventajas que aporta dicho modelo de aprendizaje frente a los clásicos y conseguir las aptitudes para poder utilizarlo.

Tras esta primera fase de introducción general a las nuevas tecnologías, se entra de lleno en la aplicación de éstas en un problema concreto, el Reconocimiento Óptico de Caracteres. Es decir, se fija el objetivo final del proyecto que consiste en realizar un sistema de reconocimiento de caracteres partiendo de un conjunto de imágenes complejas y reales proporcionadas por ArcelorMittal.

Las imágenes utilizadas en este proyecto se caracterizan por contener un solo carácter debido a la aplicación de una fase anterior de segmentación de las imágenes que captan las cámaras originalmente. La realización de esta fase de segmentación queda fuera de los objetivos fijados para el trabajo. De modo que la complejidad que reside en ellas es en primer lugar el fondo e iluminación que tienen, es decir, son imágenes en las que el color del carácter y el color del fondo es muy parecido y por lo tanto en gran parte de los casos son poco legibles y difíciles de reconocer incluso para el ser humano. Además, debido a las condiciones climáticas (lluvia) algunas contienen una gran masa de cascarilla haciendo aún más complicada la labor de distinción entre carácter y fondo.

Siendo así, se pretende desarrollar un sistema de clasificación de caracteres haciendo uso de un modelo de CNN razonado y estudiado, acorde a dicha tarea y cuyo rendimiento se asemeje a la clasificación que realizaría un ser humano. Por lo tanto, se deberá de realizar un estudio del estado del arte enfocado al reconocimiento de caracteres basado en CNNs.

Finalmente, tras dar con el enfoque de modelo apropiado, se llevará éste a la práctica, realizando un plan de experimentación en busca de posibles mejoras en el modelo, encontrando aquel que proporcione el mayor rendimiento al sistema.

En resumen, los objetivos fijados son:

- Estudio tanto teórico como práctico de las CNNs, adquiriendo los conocimientos necesarios para realizar posteriormente el desarrollando del sistema.

- Estudio del arte de la tarea de Reconocimiento de óptico de Caracteres basado en CNNs.
- Realización de una implementación completa de un sistema de reconocimiento de caracteres basado en CNNs, pudiendo obtener una clasificación de nuevas imágenes y resultados textuales y gráficos sobre el estudio realizado y los modelos encontrados en el estado del arte.
- Plan de experimentación de los modelos encontrados, seleccionando aquel que proporcione el rendimiento más alto en base a las métricas comúnmente utilizadas en este tipo de problemas.

3. Antecedentes

En este apartado se realizará una introducción al modelo de redes neuronales convolucionales o CNNs, ya que será el utilizado en la solución planteada para cumplir con los objetivos definidos anteriormente. También se realizará una descripción de las métricas de evaluación que se utilizarán para comparar los modelos analizados tras el estudio del estado del arte. Además, se detallarán los diferentes conjuntos de datos estándar utilizados en la literatura para la realización de los estudios planteados. Y por último, se describirá el sistema actual y el conjunto de datos a utilizar en este trabajo.

3.1 Redes Neuronales Convolucionales (CNNs)

Las Redes Neuronales Convolucionales engloban un modelo de aprendizaje automático que en los últimos años ha desarrollado un éxito importante debido a la consecución de resultados que eran imposibles de obtener mediante otras técnicas, siendo esta la razón por la cual su aplicación para diferentes problemas se ha extendido en la actualidad. Uno de los principales campos en los que ha adquirido notoriedad es en el de visión artificial, debiéndose a las características que presentan en su modelo dichas redes.

3.1.1 Redes Neuronales básicas

Las redes neuronales clásicas o también denominadas perceptrón multicapa. Se inspiran en las redes neuronales biológicas adquiriendo ciertas aptitudes características de ellas como:

- El procesamiento de la información ocurre en elementos simples denominados neuronas.
- Las señales son transferencias entre neuronas a través de enlaces de conexión.
- Cada conexión tiene un peso asociado.
- Cada neurona aplica una función de activación.

El modelo de neurona artificial descrito por Rumelhart y McClelland (1986) se corresponde con la Ilustración 1, consistiendo en:

- Un conjunto de entradas $x_1, \dots x_n$
- Los pesos sinápticos $w_1, \dots w_n$, correspondientes a cada entrada.
- Una función de agregación, $\sum$.
- Una función de activación, g.
- Una salida, Y.

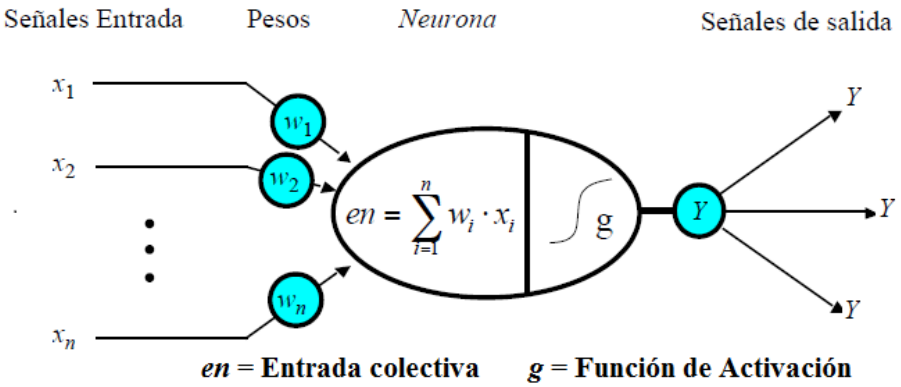

Ilustración 1: Modelo neuronal

Las redes neuronales se componen de numerosas neuronas agrupadas en base a una topología, estructura o patrón de conexionado denominado arquitectura. Las conexiones sinápticas son direccionales en un único sentido y cada agrupación de un conjunto de neuronas se denomina capa. Cada capa desempeña una acción determinada que se corresponde con una función matemática. Así, una red neuronal, es una composición de una o más capas.

Se distinguen tres tipos de capas: capas de entrada, capas de salida y capas ocultas. Una capa de entrada recibe datos o señales procedentes del entorno. Una capa de salida está conformada por las neuronas que proporcionan la salida. Y las capas ocultas que proporcionan libertad a la red neuronal para representar características del entorno que trata de modelar.

Una arquitectura de red neuronal básica sigue el modelo que se muestra en la Ilustración 2.

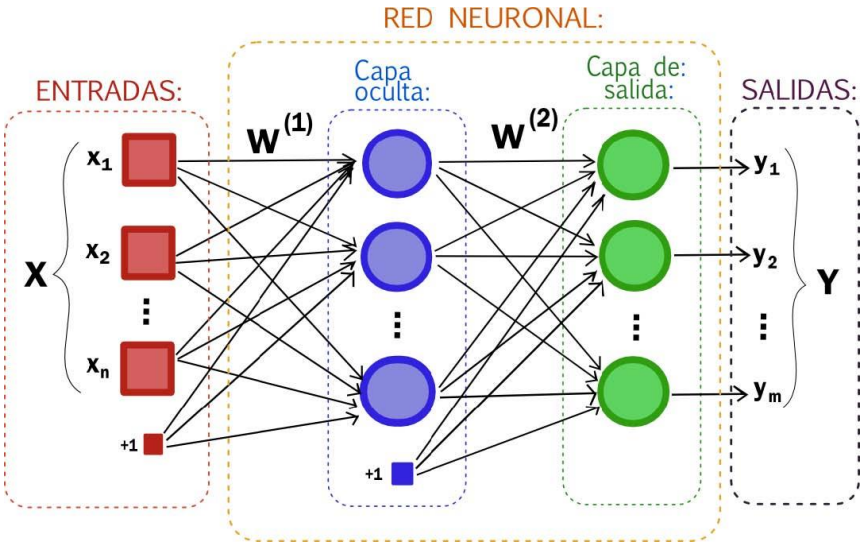

Ilustración 2: Arquitectura red neuronal

Este tipo de redes recibe como entrada y produce como salida un vector, al que en cada capa se le realiza una transformación. Además, las neuronas de cada capa están totalmente conectadas a todas las neuronas de la capa anterior. No obstante, en una capa las neuronas que la conforman no tienen conexiones siendo independientes entre sí las neuronas de una misma capa.

Esta arquitectura, sirve para datos de entrada de tamaño pequeño, pero presenta un gran problema para datos de entrada de un tamaño mayor (imágenes), de modo que, si tuviéramos imágenes en color de tamaño 128x128 pixeles, para cada neurona de la capa de entrada se deberían procesar 128x128x3 = 49 152 pesos, siendo un volumen de datos muy elevado y presentando un problema de escalabilidad de este tipo de redes para la resolución de problemas de visión artificial. Es por este motivo, que las redes neuronales convolucionales han destacado en este ámbito.

Los principales motivos por los cuales las redes neuronales convolucionales son las más indicadas para resolver problemas de visión son:

- La tipología de la estructura de entrada, en este caso acepta varios vectores 2D, pudiendo introducir uno por cada espacio de color RGB.
- Conexiones locales en lugar de capas totalmente conectadas
- Pesos compartidos entre diferentes conexiones
- Utilización de la técnica de pooling, pudiendo así reducir el número de neuronas en cada capa.
- Creación de modelos mucho más profundos al poder reducir el número de neuronas por capa.

3.1.2 Arquitectura y tipos de capas de las CNNs

Las redes neuronales convolucionales crean una arquitectura de capas en la que cada capa está compuesta por agrupaciones 3D de neuronas. Inicialmente se recibe las imágenes a tratar por las capas en la capa de entrada donde se especifica el tamaño de la agrupación 3D que la conforma, esto se corresponde con la altura, la anchura y el número de canales de color de la imagen o también llamado profundidad. Además, en esa capa inicial también puede realizarse la normalización de los datos previamente a ser procesada por las siguientes capas. Tras obtener la imagen de entrada se pueden realizar diversas operaciones siendo posible la utilización de los siguientes tipos de capas:

I. Capa de convolución

Tipo de capa que da nombre a la red, tiene como principal objetivo la extracción de características de las imágenes recibidas como entrada. Esta operación se lleva a cabo aplicando un conjunto de filtros entrenables a la imagen mediante la aplicación de máscaras, para máscaras diferentes el resultado es diferente. Los parámetros de los filtros son

aprendidos por la red para su activación al encontrar ciertas características. La máscara aplicada para cada filtro tiene unas dimensiones de altura y anchura fijadas en el diseño y una profundidad dependiente de la profundidad de la capa de entrada. Los filtros se desplazan en "pasos" cuyo valor debe de ser fijado y para cada valor de paso fijado se realiza la operación del producto escalar entre la matriz de entrada y la matriz del filtro, más la suma de un valor de sesgo, una demostración de la realización del conjunto de dichas operaciones denominadas convolución es la mostrada en la Ilustración 3.

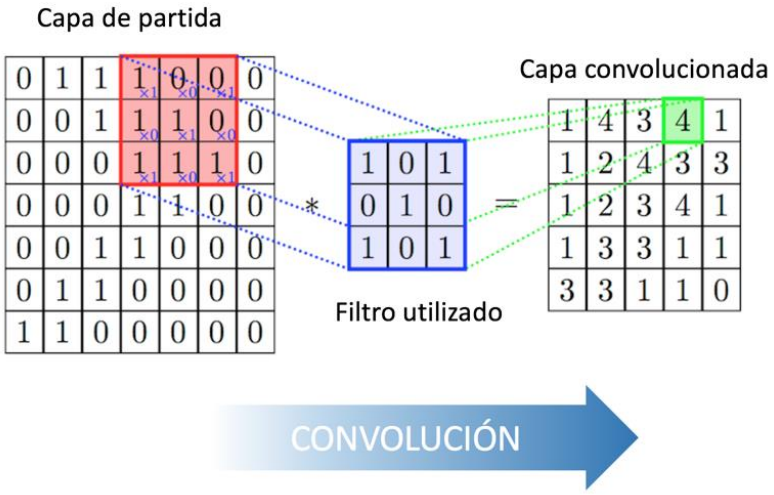

Ilustración 3: Operación realizada en la capa convolucional

Tras la utilización de una capa de convolución se aplica una función de activación con el objetivo de agregar no linealidad al modelo, eliminando la relación entre la entrada y la salida. Por eso existen las capas de activación, las cuales dependiendo de la función utilizada realizan una funcionalidad concreta.

II. Capa de activación

Como ya se ha comentado este tipo de capa se aplica tras una capa de convolución con el objetivo de agregar no-linealidad al modelo de CNN que se pretende entrenar. Los diferentes tipos existentes son: Sigmoide, Tangente hiperbólica y ReLU (Rectified Linear Unit), la más utilizada en el modelo de CNNs son las ReLU.

a. Sigmoide

$$\Phi_{\,sigmoide}\,(x) = \frac{1}{1 + \,e^{-x}} \qquad (1)$$

Esta función presentada en (1), fue inicialmente muy utilizada por su sencillez. Tiende a 1, estado de activación, para valores altos de x y a 0, estado de no activación, para valores negativos, siendo problemáticos los valores extremos donde la derivada es 0.

b. Tangente hiperbólica

$$\phi_{\tanh(x)} = \tanh(x) = 2\phi_{sigmoide}(2x) - 1 \quad (2)$$

Similar a la función Sigmoide, (2) difiere en el rango de valores que toma, en este caso de -1 a 1. Al igual que la anterior actualmente se utiliza poco.

c. ReLu

$$\phi_{ReLu} = \max(x, 0) \quad (3)$$

Utilizada en casi la totalidad de modelos de CNNs actuales (3), se probó su gran rendimiento en [Krizhevsky2017], en gran parte por la gran reducción de complejidad respecto a las anteriores, ya que se trata de un máximo. Solo tiene un punto problemático en $x = 0$, por lo tanto, el problema es menor que el de las funciones anteriores.

III. Capa de pooling

En esta capa se aplica un algoritmo para reducir las dimensiones de la capa entrante preservando la información más relevante y eliminando aquella que no lo sea, además de disminuir el volumen de datos a procesar y por tanto el tiempo de procesamiento del entrenamiento. Existen diferentes enfoques para realizar la supresión de información irrelevante, generalmente la más usada es Max pooling, la cual se queda con el valor más alto de la subregión tratada como se puede ver en la Ilustración 4. Otra de las más usadas es Average pooling, en la que se calcula el valor medio de cada subregión, siendo este el valor utilizado.

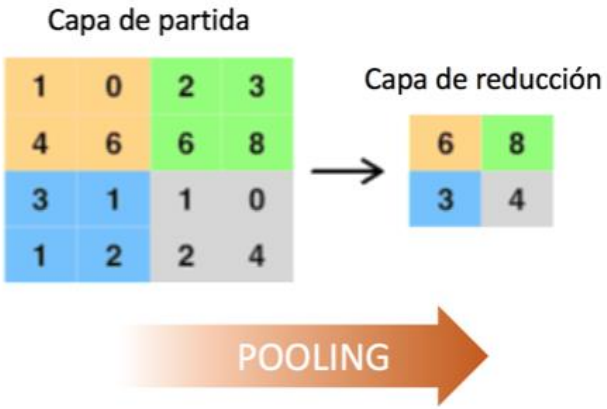

Ilustración 4: Operación Max pooling

IV. Capa totalmente conectada

Capa que realiza la tarea de clasificación basándose en las características extraídas por las capas precedentes. Cada neurona que forma la capa está completamente conectada con las

neuronas de salida. Este tipo de capas aprenden un vector de pesos que asignan a cada conexión entre una neurona de entrada con una de salida y un vector de sesgo. Su objetivo es aproximar la función objetivo.

V. Capa de salida

Capa que completa la fase de clasificación de la red neuronal convolucional, normalmente va a continuación de una capa totalmente conectada, con el objetivo de completar la clasificación obteniendo una salida acorde a la definición del problema. Es decir, si la tarea a realizar pretendía clasificar en n clases, la salida de esta capa consta de n neuronas, que contienen la probabilidad de cada imagen de pertenecer a cada clase. Esto se realiza mediante una función de transformación, para la tarea de clasificación la función utilizada generalmente es la Softmax definida en (4). Devuelve n valores comprendidos entre [0,1] que representan la probabilidad de pertenecer a dicha clase. La suma de los n valores es 1.

$$Softmax(x) = \frac{e^{x_1}}{\sum_{j-1}^{n} e^{x_j}} \tag{4}$$

3.1.3 Otros conceptos importantes utilizados en CNNs

I. Backpropagation

Backpropagation o la propagación hacia atrás es un método para entrenar de manera automática las redes neuronales con capas ocultas. Se parte de tener como conocimiento el valor de la capa de entrada y la capa de salida, de modo que no se pueden ajustar los pesos sinápticos asociados a las neuronas que forman la capa oculta. Por lo tanto, lo que se busca son los valores de los pesos sinápticos de las conexiones entre esas capas. Para ello, el procedimiento que se sigue es: elegir de manera aleatoria unos datos de entrada que formen parte del conjunto de entrenamiento y pesos sinápticos también aleatorios y dejar que la red genere una salida. Luego se realiza una comparación entre la salida obtenida de manera aleatoria y la salida que se conoce, la diferencia entre estas dos se denomina error de y es usado para ajustar los pesos sinápticos de la capa de salida. Tras la obtención de este error, lo que se hace es propagar el error hacia las neuronas de la capa anterior y ajustar los pesos sinápticos de esta, así hasta alcanzar la capa de entrada. Este concepto fue redescubierto y formalizado y explicado en [Rumelhart1986]

II. Overfitting

Se trata de un problema presente en aquellas redes en las que el error obtenido en la fase de entrenamiento dista en gran medida del error de generalización. Es decir, si en la fase de entrenamiento de la red el error es pequeño y al realizar la fase de test se obtienen valores de error altos se produce overfitting o sobreajuste, debido a que la red memoriza demasiado los patrones de características del conjunto de entrenamiento y es incapaz de generalizar a

nuevos patrones similares, obteniendo así una mala generalización de la red. Para poder solventar dicho problema, existen dos posibilidades. Por un lado, proporcionar más instancias de datos, incrementando de este modo el conjunto de datos utilizados para el entrenamiento y consiguiendo que no se produzca un entrenamiento excesivo o sobre aprendizaje. Por otro lado, se puede reducir la complejidad de la red que realiza el aprendizaje, así con el mismo número de instancias de entrenamiento se puede reducir el sobre aprendizaje y obtener una generalización mejor de la red.

III. Dropout

Se trata de una técnica de regularización de la red neuronal convolucional mostrada en la Ilustración 5. Dicha técnica consiste en la desconexión de un porcentaje de neuronas de la red en cada iteración del entrenamiento mejorando de este modo la generalización de la red ya que se evita que las neuronas se adapten, es usada en grandes redes donde es difícil hacer frente al overfitting que pueden tener. En [Srivastava2014] se realiza un estudio y experimentación del uso de dicha técnica, mostrando la mejora de rendimiento de las redes neuronales para varias aplicaciones.

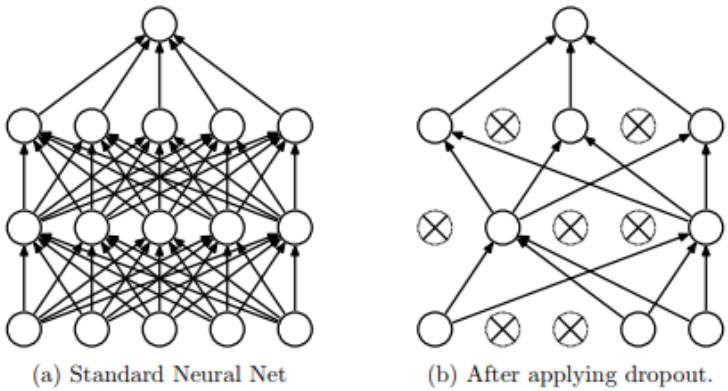

Ilustración 5: Aplicación de dropout en una red neuronal

IV. Optimizador

En la fase de entrenamiento de las redes neuronales convolucionales se utiliza una función de optimización que en función de los parámetros y restricciones introducidas busca obtener le mejor rendimiento de la red. La función de optimización tradicional es la denominada ***Stochastic gradient descent*** o ***descenso de gradiente estocástico*** (SGD), que con el tiempo ha dejado de ser tan usada debido a la mejora del rendimiento de los modelos con la función Adam (A method for stochastic optimization descrita en [Kingma2014]).

3.2 Métricas de evaluación

Para poder realizar una labor de análisis y comprensión de los distintos métodos presentados en la literatura y realizar un proceso de evaluación de la solución propuesta en este trabajo, se necesita conocer una serie de métricas de evaluación, entenderlas e interpretar de manera correcta los resultados. Es por eso, que a continuación se realizará una breve descripción de las más importantes.

3.2.1 Métricas clasificación binaria

La evaluación a realizar en este trabajo requiere que el conjunto de datos este etiquetado previamente de manera correcta, denominándose cada instancia etiquetada como valor real, mientras que la predicción realizada por el clasificador de cada instancia se denomina valor predicho. Las métricas binarias, son aquellas que tienen en cuenta dos categorías para cada tipo de valor (real o predicho), la categoría positiva P y la categoría negativa N. De este modo, el valor obtenido por el clasificador puede ser:

- **TP (True positive**): El valor real y el valor predicho por el clasificador son ambos positivos.
- **FP (False positive):** El valor real es negativo, pero el clasificador lo ha predicho como positivo.
- **TN (True negative):** El valor real y el valor predicho por el clasificador son ambos negativos.
- **FN (False negative):** El valor real es positivo, pero el clasificador lo ha predicho como negativo.

Por lo tanto, el clasificador realiza dos tipos de predicciones correctas y dos tipos de errores, que pueden ser representadas mediante una matriz de confusión como la mostrada en la Ilustración 6.

Confusion matrix		**Actual class**	
		Positive	Negative
Predicted class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Ilustración 6: Matriz de confusión

3.2.2 Exactitud (accuracy), precisión, recall y especificidad (sprecifity).

I. Accuracy

Métrica muy utilizada que mide la fracción de predicciones correctas respecto al total de las instancias, como se define en (5)

$$Accuracy = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5)$$

No obstante, dicha métrica presenta un problema para conjuntos de datos en los que la proporción de instancias de cada clase sea desbalanceada. Es por este motivo, que se usarán otras métricas cuya información sea más confiable.

II. Precisión

Mide la fracción de positivos reales entre los ejemplos que se predicen como positivos como se define en (6).

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

III. Recall

Indica la fracción de instancias positivas correctamente clasificadas. Por este motivo también puede llamarse true positive rate y se define en (7).

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN} \quad (7)$$

IV. Especificidad (specifity)

Indica la fracción de instancia negativas cuyo estado real es también negativo. Se define en (8).

$$Specifity = \frac{TN}{N} = \frac{TN}{TN + FP} \quad (8)$$

Como se puede ver, todas estas métricas parten de los valores obtenidos en las métricas binarias TP, TN, FP y FN. Siendo representado como se muestra en la Ilustración 7.

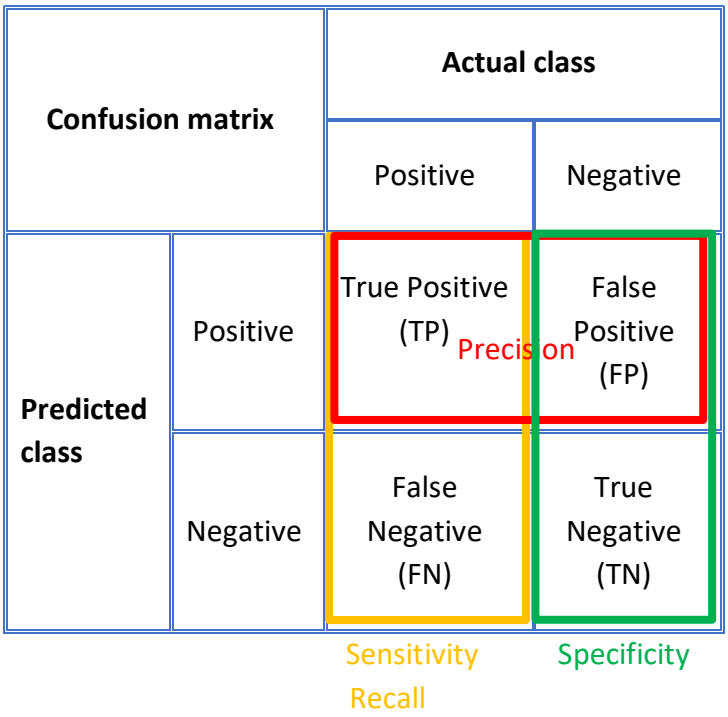

Ilustración 7: Representación métricas relevantes respecto a métricas binarias

3.2.3 F_1-Score

Se trata de la media armónica de la precisión y el recall, es decir, se le da el mismo peso al recall que a la precisión, obteniendo una comparación del rendimiento de dos clasificadores de modo más sencillo. Se define en (9).

$$F_1 = 2\frac{Precision \cdot Recall}{(Precision + Recall)} \tag{9}$$

3.2.4 Curva ROC y AUC

Una Curva ROC (Receiver Operating Characteristic) es una representación gráfica de la sensibilidad (recall) en función del ratio de falsos positivos (1- Especificidad) para distintos umbrales que se utilizan para clasificar un ejemplo como positivo o negativo. Obteniendo así, una representación global de la exactitud diagnóstica. Como se puede ver en la Ilustración 8, es una curva necesariamente creciente. El área bajo esta curva se denomina AUC (Area Under a ROC Curve) y se trata de una medida de calidad que mide la capacidad del modelo de predecir una mayor puntuación para instancias positivas en comparación con instancias negativas. Es independiente del umbral seleccionado y cuanto mayor sea el área mejor es el clasificador.

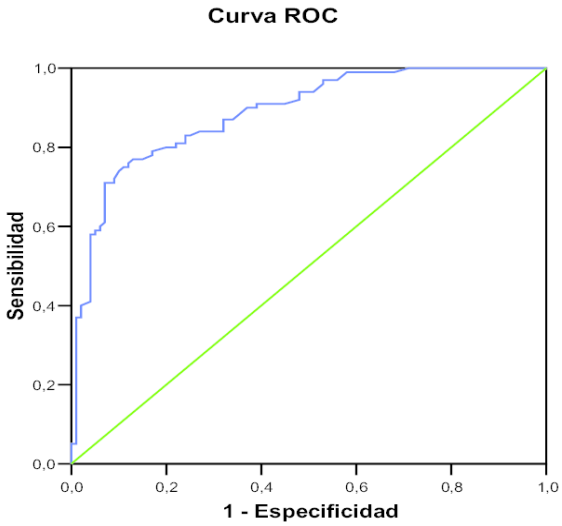

Ilustración 8: Curva ROC

3.3 Conjuntos de datos estándar

Generalmente, los conjuntos de datos utilizados para la realización de pruebas y experimentos en los sistemas propuestos en la literatura son recurrentes. Esto se debe a que los sistemas diseñados varían en gran medida dependiendo del problema al que se busque una solución, y así, a lo largo de los años se han ido creando conjuntos de datos para los distintos problemas a los que se buscaba una solución. Por lo tanto, en este apartado se realizará una breve descripción de los dataset estándar a los que se va a hacer referencia en el estudio previo teórico sobre reconocimiento de caracteres en imágenes en las que aparezca un solo carácter.

3.3.1 MNIST

Es el dataset por referencia utilizado en todos los artículos relacionados con el reconocimiento de dígitos, encabezado por el científico Yann LeCun, uno de los pioneros en el campo de las redes neuronales convolucionales aplicadas a problemas de visión artificial en 1998.

Es un conjunto de imágenes de tamaño 28x28 en blanco y negro. Como se puede observar en Ilustración 9, cada imagen del dataset contiene un dígito del 0 al 9 escrito a mano en color negro sobre fondo blanco. Está compuesto por dos subconjuntos, uno de 60.000 instancias utilizado para la fase de entrenamiento y otro de 10.000 instancias para la realización del test.

Este conjunto ha sido extraído del dataset NIST. Concretamente está compuesto por los dataset especiales 1 y 3, llamados SD-1 y SD-3, estos dataset no tienen una distribución libre,

solo se hace referencia a ellos en la web oficial del NIST, donde existen muchos otros datasets que si son de distribución libre. De este modo, tras la mezcla de los dataset SD-1 y SD-3 construyendo el conjunto de imágenes que conforma el dataset MNIST, se les aplicó a los dígitos un proceso de normalizado, de modo que, el nivel del fondo (blanco) se corresponda con un valor de -0.1 y los dígitos (negro) con un valor de 1.175, así cada dígito contiene suficientes características que lo hagan fácilmente reconocible. Además, cada dígito normalizado ha sido centrado en la imagen resultante.

El dataset está enfocado para la introducción en el aprendizaje de técnicas y métodos de reconocimiento de patrones con datos del mundo real sin realizar gran esfuerzo en una fase de procesamiento o formateo de datos.

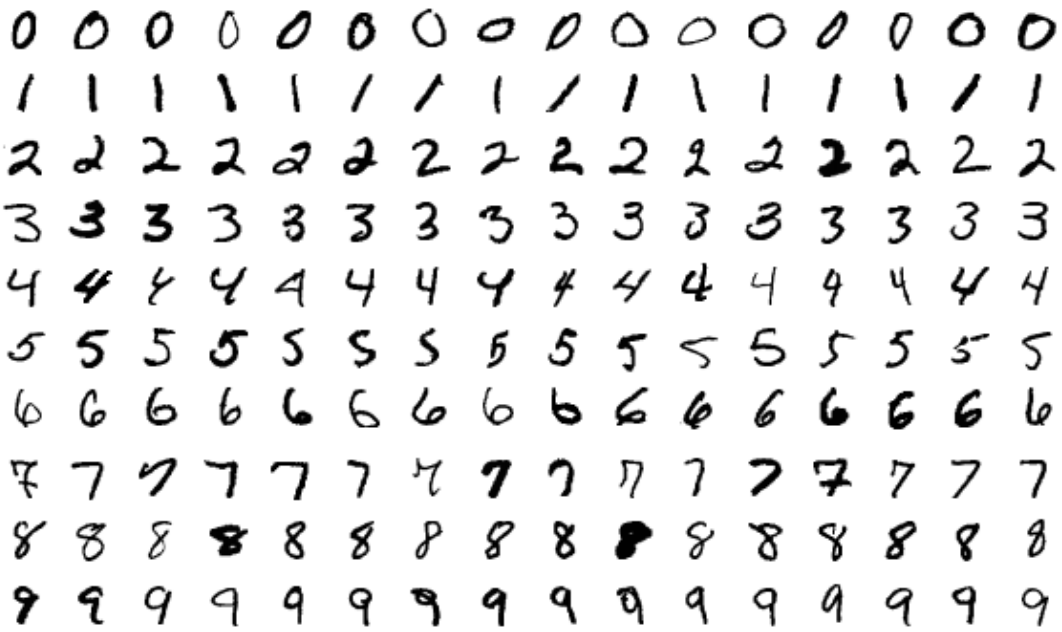

Ilustración 9: Ejemplo de imágenes que conforman el dataset MNIST

3.3.2 ICDAR 2003

El conjunto de datos ICDAR 2003 [Lucas2003], fue creado debido a la necesidad de establecer unos datos de referencia cuyo texto no fuese reconocido por paquetes OCR comerciales. De este modo, se podría probar y comparar los sistemas diseñados para dar solución a la tarea de reconocimiento de caracteres en imágenes cuyas propiedades sean complejas.

En función del alcance del problema al que se quiere encontrar solución se pueden encontrar tres datasets distintos. Estos se corresponden con tres subproblemas del problema global de reconocimiento. Es decir, existe un dataset cuyas imágenes están pensadas para realizar tareas de localización de texto. Las imágenes que conforman este dataset se tratan de

escenas reales que contienen texto, de ahí que se halla denominado scene. Un ejemplo de este dataset es el mostrado en la Ilustración 10.

Ilustración 10: Imagen del dataset scene de ICDAR 2003

Otro de los dataset es el denominado Word. Sus imágenes están pensadas para realizar la tarea de reconocimiento de palabras. Un ejemplo de éstas es el mostrado en Ilustración 11.

Ilustración 11: Imagen del dataset word de ICDAR 2003

Por último, para el reconocimiento de caracteres se utiliza el dataset char, siendo las imágenes similares a la Ilustración 12.

Ilustración 12: Imágenes de dataset char de ICDAR 2003

Cada conjunto de datos se proporciona como un archivo .zip que contiene imágenes jpeg y un archivo .xml con las etiquetas.

Para la realización de este trabajo se hará referencia al dataset char, debido a que las imágenes que se pretende reconocer contienen caracteres individuales y los estudios valorados hacen uso de dicho conjunto. El conjunto está dividido en tres subconjuntos: el subconjunto de entrenamiento formado por 6185 imágenes, el subconjunto de test

compuesto por 5430 imágenes y el subconjunto de validacion compuesto por 854 imágenes. Las dimensiones de las imágenes son variables, habiendo imágenes de tamaño 5x12, 36x47 y 206x223. Además, tiene distintos colores, fondos y distorsiones.

4. Sistema actual de captura, preprocesamiento y reconocimiento de imágenes

A continuación, se describirá la situación de partida del trabajo en cuanto a las imágenes de entrada que se van a utilizar y el procedimiento de segmentación que se sigue para obtenerlas partiendo de un conjunto de imágenes completas iniciales. Además, también se describirá de manera resumida el sistema que se utiliza para realizar la captura de las imágenes y el sistema que las procesa realizando el reconocimiento de caracteres, así como algunas de las apreciaciones que se han obtenido para explicar el rendimiento que se obtiene con él.

El sistema está compuesto por una infraestructura física para la captura de las imágenes y un sistema software encargado de realizar la labor de reconocimiento de los caracteres.

4.1 Captura de imágenes

Las imágenes que se capturan mediante la infraestructura física son partes de carriles en las que se encuentran impresas cadenas de caracteres. Cada cadena de caracteres se corresponde con un troquel e identifica un carril.

Para llevar a cabo la adquisición de las imágenes se tiene desplegado un sistema de hardware compuesto por un soporte por el que se mueve el carril y en diferentes ángulos de inclinación respecto a dicho soporte, tres cámaras para realizar las fotografías del carril, cada una de ellas con una fuente de luz distinta, siendo los colores que proyectan: azul, rojo y verde. Este sistema de color se basa en la descomposición de colores del espacio RGB.

De este modo el funcionamiento del sistema se basa en el avance del carril por el soporte con una velocidad determinada y en una dirección constante mientras las cámaras van tomando fotos de un lado del carril mientras éste avanza, capturando de este modo instantáneas de la totalidad del carril, tanto aquellas zonas en las que hay caracteres como las zonas de carril sin caracteres, siendo estas las que identifican el fin de una cadena de identificación y por consiguiente de un troquel y el comienzo de otro

Las imágenes obtenidas por cada cámara en un mismo instante de tiempo y velocidad son unidas obteniendo una imagen final producto de la composición de las tres cámaras. De este modo, se obtiene una imagen final con los tres canales de color.

Cada cadena de caracteres es iniciada por imágenes sin caracteres, seguida de imágenes en las que aparecen los caracteres, en algunos casos aparece el mismo carácter en varias imágenes, situación que será identificada gracias a la información de desplazamiento, y finalmente, cuando acabe la cadena volverán a aparecer imágenes sin caracteres. De este modo se van creando secuencias de imágenes que se corresponden con una cadena. El sistema actual realiza varios procedimientos para reconocer la cadena completa.

Una muestra de las imágenes obtenidas por dicho sistema son las que se puede ver en la siguiente secuencia de imágenes: Ilustración 13, Ilustración 14,Ilustración 15, Ilustración 16, Ilustración 17, Ilustración 18, Ilustración 19 e Ilustración 20. En ella se puede visualizar como se captura una cadena de caracteres completa en varias imágenes. Dicha cadena que identifica cada troquel sigue una nomenclatura, basada en 14 caracteres, de los cuales todos son dígitos a excepción del carácter impreso en la posición 11, que se trata de una letra en mayúscula.

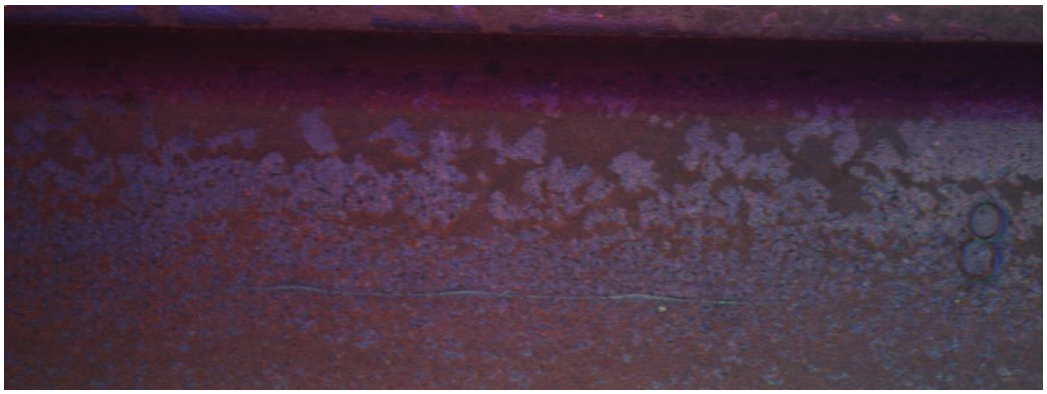

Ilustración 13: Imagen 1 de la secuencia

Ilustración 14: Imagen 2 de la secuencia

Ilustración 15: Imagen 3 de la secuencia

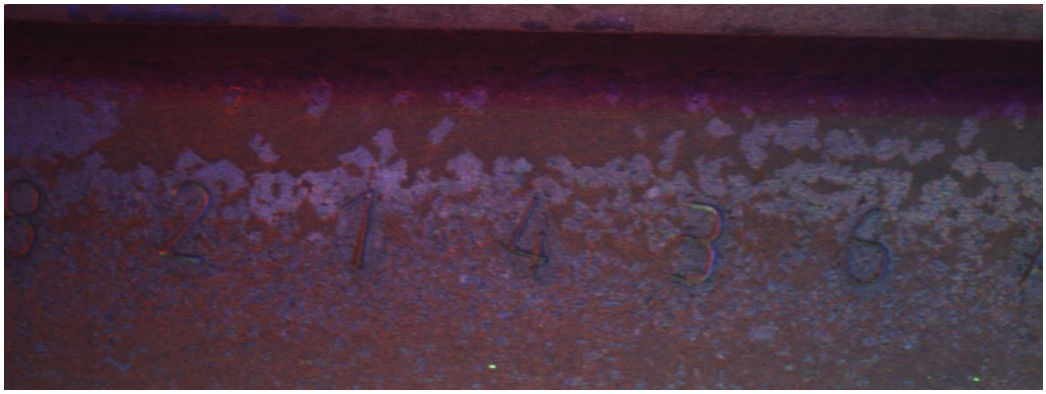

Ilustración 16: Imagen 4 de la secuencia

Ilustración 17: Imagen 5 de la secuencia

Ilustración 18: Imagen 6 de la secuencia

Ilustración 19: Imagen 7 de la secuencia

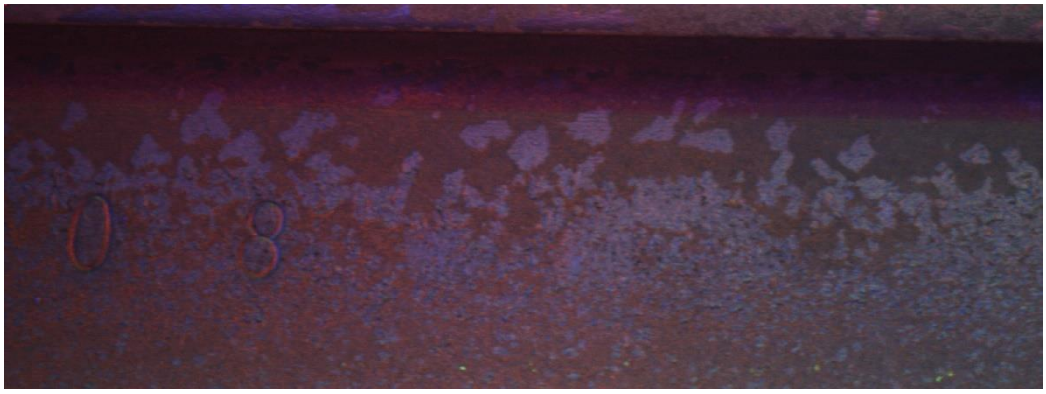

Ilustración 20: Imagen 8 de la secuencia

Finalmente, como se ha comentado también se capturan imágenes en las que no hay presente ningún carácter impreso, ese es el caso de la Ilustración 21.

Ilustración 21: Imagen sin caracteres

4.2 Segmentación de las imágenes

Las imágenes capturadas por el sistema de cámaras descrito anteriormente presentan unas características de complejidad muy alta debido a que contienen más de un carácter y a que las condiciones de legibilidad, iluminación, color, cascarilla presente, etc., no son las deseables. Por lo que lo convierte en un problema de reconocimiento de caracteres en imágenes de escenas reales, un tipo de problema que en la literatura actual basada en CNNs aún no ha sido resuelto por ningún sistema con una precisión similar a la que desarrolla un ser humano. Esto se debe a que se trata de un problema compuesto para el que hay que realizar dos tareas encadenadas. En primer lugar, una fase de detección de caracteres, obteniendo las zonas exactas de la imagen donde aparecen los caracteres y realizar la segmentación de ellas para obtener los caracteres individuales. Para a continuación realizar la segunda parte en la que se realiza el reconocimiento o clasificación de los caracteres recortados. Esto presenta una dificultad añadida para las imágenes que se tienen en este caso, y es que contienen cascarilla que dificulta la tarea de detección, obteniendo un resultado de esa fase en la que se pierde gran rendimiento y penaliza mucho el resultado final.

Es por eso, que se ha realizado la fase de segmentación mediante técnicas de visión artificial tradicionales como se ha realizado hasta el momento.

El procedimiento se realiza mediante el software Halcon, y se ha implementado mediante un script en el que en base a los tres canales del espacio RGB se obtiene la profundidad de cada carácter y se realiza la segmentación de dicho carácter. Tras este procedimiento las imágenes resultantes son las que se pueden ver a continuación en la Ilustración 22.

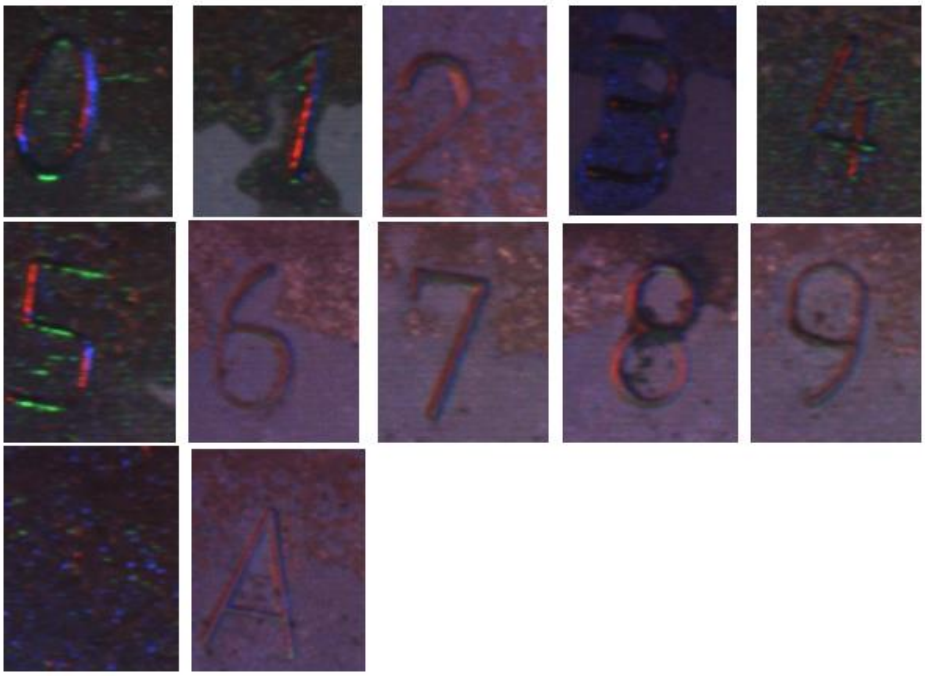

Ilustración 22: Imágenes resultantes del proceso de segmentación.

Las imágenes recortadas serán las utilizadas para entrenar los diferentes modelos de CNNs. La organización de estas imágenes es la siguiente: estan divididas en directorios según el carácter que contengan, de modo que los digitos están almacenados en carpetas cuyo nombre equivale al propio dígito del 0 al 9, las imágenes que no contengan ningún carácter estan almacenadas en el directorio 10 y las imágenes que contengan una letra estan almacenadas con el siguiente número consecutivo, es decir, la letra A es el 11, la B es el 12 y asi hasta el número 36 correspondiente a la Z.

Las imágenes tienen una extensión .TIF y dimensiones con pequeñas variaciones siendo algunas de tamaño 145x186, 145x185, 145x177, 145x176, etc..

4.3 Sistema de reconocimiento

Tras realizar la labor de segmentación actualmente se sigue un procedimiento para realizar el reconocimiento de una cadena de caracteres correspondiente a una secuencia completa como la mostrada en las ilustraciones de la 13 a la 20. Esto lo consiguen mediante la información del desplazamiento de cada imagen segmentada, que a su vez se obtiene gracias a la información de tiempo y velocidad a la que circula el carril y en la que se obtiene cada imagen. Es decir, se va componiendo la cadena de caracteres en base a las imágenes segmentadas y a la información de desplazamiento, sabiendo la posición que tienen en la secuencia todos los caracteres, pero sobre todo, siendo importante para aquellos caracteres que aparecen en varias imágenes, ya que con esta información se pueden distinguir. De este modo se vuelve a recomponer la cadena de caracteres.

El problema que presenta el sistema es la gran rigidez del sistema de reconocimiento basado en el cálculo de la profundidad de los caracteres, debido a la cascarilla presente en las imágenes que genera gran ruido y confunde al sistema y también debido a la no normalización de las imágenes, pudiendo a veces aparecer el carácter en distinta ubicación dentro de la imagen. Además, el desbalanceo de las distintas clases al utilizar un umbral de confianza de clasificación bastante alto, hace que los caracteres que aparecen de manera menos frecuente sean clasificados como caracteres de clases que aparecen con más frecuencia.

Por lo tanto, todos estos factores hacen que sea necesario la utilización de tecnologías más flexibles como son las CNNs.

5. Estudio del estado de la técnica

En el presente apartado se realizará un estudio sobre los diversos sistemas existentes en la literatura en los que se hace uso de redes neuronales convolucionales para solucionar la tarea de reconocimiento de caracteres en imágenes que contienen un solo carácter.

A lo largo de las últimas décadas, el reconocimiento de caracteres (OCR) ha sido abordado por gran número de investigadores debido a su gran importancia para la digitalización de documentos, obteniendo unos resultados muy satisfactorios. No obstante, las imágenes que se han facilitado para este trabajo poseen características de fondo, iluminación, legibilidad y nitidez notoriamente más complejas, situación que aumenta la complejidad de la tarea de reconocimiento y la cual ha impulsado la realización de un estudio teórico del problema.

Como consecuencia de estos hechos y teniendo en cuenta que nuestro dataset está compuesto por imágenes segmentadas en las que aparece un único carácter por imagen, se decidió dividir el estudio del arte en dos partes. Comenzando el estudio por aquellos sistemas basados en CNNs que realicen el reconocimiento de un carácter, en el caso concreto y más sencillo en el que dicho carácter se trata de un dígito, para luego continuar con aquellos estudios en los que se realice el reconocimiento de todos los caracteres alfanuméricos.

Esta decisión, fue tomada en vista de que la mayoría de los caracteres que aparecen en las imágenes proporcionadas para llevar acabo el reconocimiento se corresponden con el conjunto de dígitos, interpretando que este sería un punto de partida razonable para resolver el problema y poder probar que resultados se obtendrán para un conjunto mayor compuesto por la totalidad del alfabeto. Y así, partiendo de dicho punto, poder agregar las prácticas necesarias para poder extender el conjunto y que los resultados sean los esperados para el reconocimiento del conjunto total.

5.1 Estado de las técnicas de reconocimiento de dígitos basado en CNNs

La tarea de reconocer dígitos mediante la utilización de modelos de CNNs ha experimentado una gran evolución en las últimas décadas, obteniendo un rendimiento muy similar al humano. A continuación, se realizará un recorrido de la evolución generada, describiendo algunos de los estudios que han impulsado la obtención de dichos resultados.

5.1.1 Gradient-Based Learning Applied to Document Recognition

Se trata de un estudio pionero que combinó el modelo de CNNs y la visión artificial para realizar el reconocimiento de dígitos [Lecun1998]. En dicho estudio se analiza el rendimiento que proporciona utilizar el modelo de CNNs junto con otras técnicas de entrenamiento

obteniendo unos resultados de récord frente a la utilización de otros métodos de diseño manual como los heurísticos.

El núcleo de este estudio es la utilización de las CNNs junto al aprendizaje basado en la utilización del gradiente, diseñando así un sistema único al que se denominó *Graph Transformer Networks (GTN)*. Este sistema desbancó a la tradicional manera de construir un sistema de reconocimiento que precisaba la integración manual de los módulos individuales, mientras que GTN, permitía entrenar todos los módulos para optimizarlos con un criterio de desempeño global.

Dentro de este sistema, la arquitectura de CNN que fue utilizada para realizar los experimentos y demostrar el rendimiento que proporcionan las CNNs es la denominada LeNet-5. Esta red como se puede observar en la Ilustración 23 está compuesta por 7 capas.

Una primera capa convolucional C1 que recibe una imagen de entrada de 32x32 pixeles, esta capa está compuesta por 6 filtros con un kernel de tamaño 5x5, dando como resultado de salida 6 mapas de características de tamaño 28x28 pixeles. Esta salida será la entrada para la siguiente capa de pooling S2, en la que se aplica la función average pooling (AP) haciendo uso de un kernel de 2x2. Entre estas capas se integra una capa de activación con el objetivo de agregar no-linealidad, en este modelo se utilizó una función sigmoide descrita en (1). La siguiente capa de convolución C3 aplica 16 filtros con un kernel de 5x5, dando lugar a 16 mapas de características de tamaño 10x10 a los que se aplica de nuevo una capa de pooling S4 idéntica a S2. En la capa C5 se aplican 120 filtros cuyo kernel tiene tamaño 5x5 a una entrada de 16 mapas de características de tamaño 5x5. A continuación, se añade una capa completamente conectada F6 de 84 neuronas, seguida de otra capa completamente conectada con tantas neuronas como número de clases se quiera clasificar. En el caso de MNIST son 10 clases. La función utilizada para obtener la capa de salida del modelo es una tangente hiperbólica escalada descrita en (2).

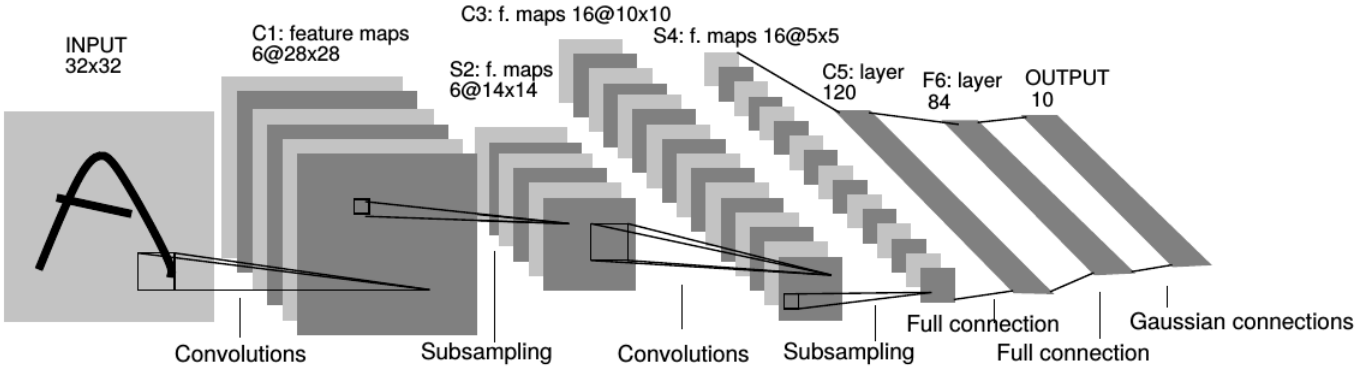

Ilustración 23: Arquitectura de red LeNet-5

[Lecun1998] utilizó como dataset de experimentación a MNIST, una modificación del dataset NIST realizada expresamente para este estudio y que se convertiría a posteriori en uno de los dataset más utilizados para realizar los estudios de reconocimiento óptico de caracteres basado en dígitos.

Los resultados de experimentación obtenidos con la red LeNet-5 fueron de un 0.95% de tasa de error en la predicción realizada para el dataset MNIST sin modificaciones, superando de manera holgada los resultados obtenidos con otras técnicas y tipos de redes como se puede observar en la Ilustración 24.

No obstante, en la Ilustración 24 también se puede observar que se probaron otros pares {dataset, arquitectura} con los cuales se obtuvo resultados mejores. Uno de ellos se trata de la misma arquitectura LeNet-5, no obstante, el conjunto de datos presenta modificaciones respecto del original MNIST, introduciendo imágenes a las cuales se realizaron distorsiones en los dígitos, esta técnica se tradujo en la reducción de la tasa de error obteniendo un valor del 0.8%.

El otro experimento realizado en el cual se obtienen resultados satisfactorios consigue una tasa de error del 0.7%. En este caso el experimento se realiza con el dataset distorsionado y además con una arquitectura distinta. Se utiliza el modelo de red LeNet-4, un modelo anterior a la arquitectura LeNet-5 y cuyas diferencias residen en el número de capas y el número de filtros utilizados en las capas de convolución. Es decir, LeNet-4 está compuesta por 6 capas, ya que no cuenta con la capa C5 de la red LeNet-5, además en la capa C1 solo utiliza 4 filtros en vez de 6. Salvando esas diferencias, las restantes capas tienen las mismas características.

Estas simplificaciones convierten a LeNet-4 en peor red con los datos originales, obteniendo 1.1% de error. Sin embargo, realizando una técnica de Boosting, basada en la combinación de los resultados de varios clasificadores, se obtienen resultados más precisos. Concretamente, en este caso se hace uso de la salida de tres clasificaciones realizadas con la arquitectura LeNet-4, que al ser sumadas disminuyen la tasa de error hasta el 0.7%

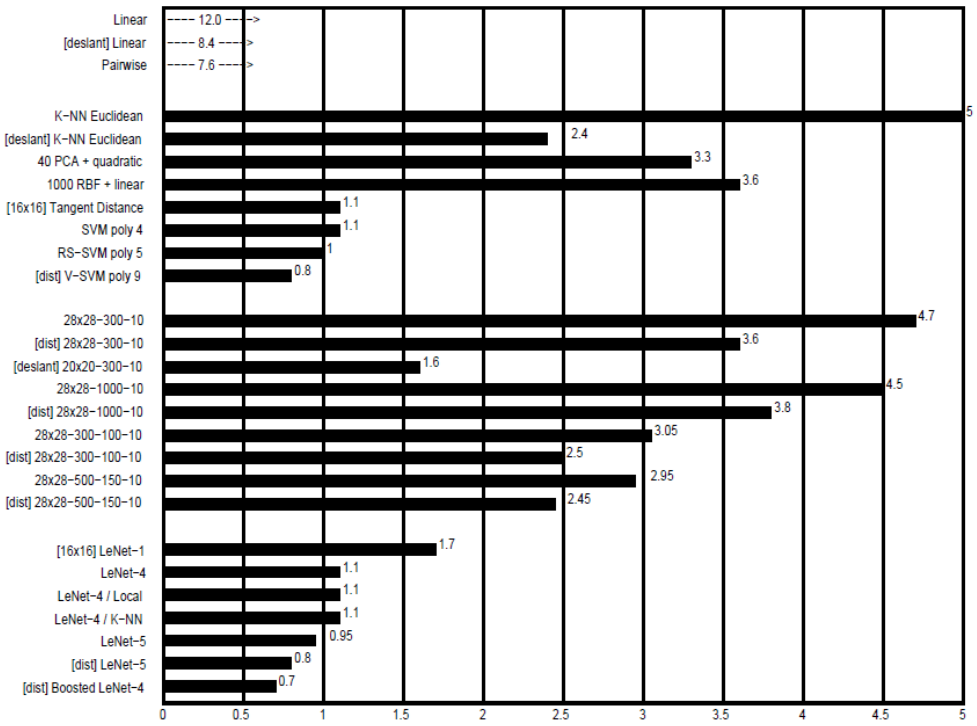

Ilustración 24: Tasa de error de la experimentación con LeNet-5 frente a otras técnicas

Gracias al punto de partida definido en el estudio [Lecun1998] la utilización de las CNNs para el reconocimiento de caracteres y en su defecto de dígitos se convirtió en el principal método estudiado e investigado, obteniendo de este modo resultados muy buenos y mejorando los resultados no solo para este problema, sino para otros más complejos derivados de éste.

Por lo tanto, se siguió indagando sobre los modelos de CNNs y otros métodos que se podrían introducir en éstos para poder mejorar los resultados, dando lugar a un amplio estudio posterior.

5.1.2 Handwritten Digit Recognition using Convolutional Neural Networks and Gabor filters

En el estudio realizado en [Calderon2003] se describe una solución al mismo problema de reconocimiento de dígitos basado en el dataset MNIST. En este caso, se utiliza una arquitectura de red ligeramente más sencilla. Compuesta por la capa de entrada, 5 capas intermedias y una capa de salida. Pero la innovación es la introducción del uso del filtro Gabor como mapa de características en la primera capa convolucional del modelo, dando lugar a una nueva topología de red denominada GCNN, además, hace uso de un algoritmo basado en el gradiente para adaptar el peso de las otras capas usadas.

Los detectores de características estándares pueden ser útiles tanto para una parte de la imagen como para su totalidad debido al uso de vectores de pesos idénticos para diferentes campos. Además, la realización de manera sucesivas de capas de convolución y pooling

intercaladas supone el aumento del número de mapas de características a la vez que disminuye la resolución espacial en comparación con la capa anterior. Reduciendo la cantidad de características, la capacidad de cómputo de la máquina utilizada y además reduciendo la diferencia entre el error de entrenamiento y el error de la validación.

Por su parte los filtros de Gabor son usados para análisis multi-resolución, ya que describen una imagen en diferentes niveles de frecuencia, permitiendo realizar una búsqueda jerárquica de objetos en la imagen y pudiendo extraer diferentes características en función de la respuesta obtenida por el filtro y la frecuencia.

La arquitectura de la red GCNN presentada en la Ilustración 25, está compuesta por las capas G1, S2, C3, S4, C5 y F6. La capa inicial G1, recibe como entrada una imagen de 28x28 a la que se le aplica los filtros de Gabor haciendo uso de 12 mapas de características, 2 frecuencias diferentes y 6 orientaciones diferentes. Las capas de submuestreo S2 y S4, utilizan una función average pooling y un tamaño de kernel de 2x2. Las capas de convolución C3 y C5 aplican 16 y 120 filtros respectivamente con un kernel de tamaño 5x5 y la capa F6 es completamente conectada y usa 84 neuronas. Además, se usa una función sigmoide para realizar la activación tras cada capa de convolución. Por último, para la activación final del modelo se utiliza la función tangente hiperbólica.

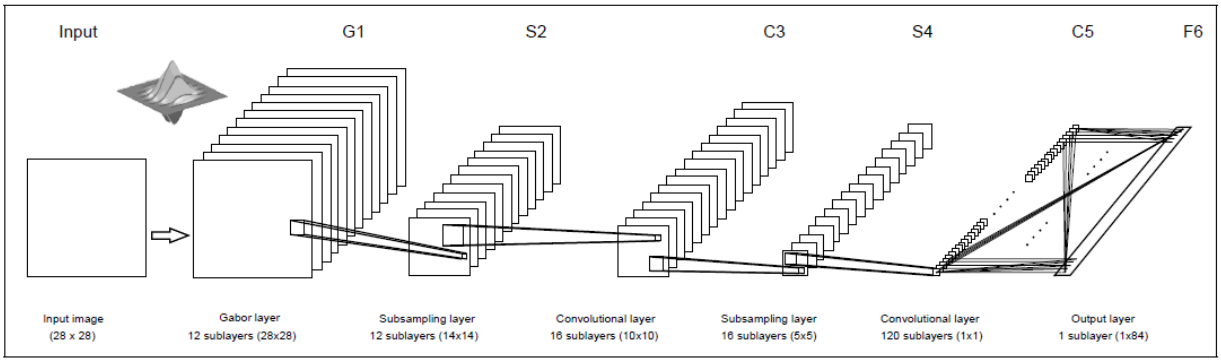

Ilustración 25: Arquitectura de la red GCNN

La fase de experimentación realizada en [Calderon2003] introduce un nuevo método de predicción denominado Boosting. Este método se basa en la combinación de varios experimentos para obtener un resultado mejor, debido a que se distribuye la tarea de aprendizaje entre los distintos experimentos. La idea principal de experimentación consiste en la combinación de predicciones precisas y otras más inexactas.

De este modo, en [Calderon2003] se realizaron tres experimentos individuales que luego serían combinados para obtener el resultado final definitivo del estudio. El primero se realizó usando la red GCNN propuesta con el conjunto de entrenamiento de MNIST y con 12 epochs, obteniendo una tasa de error de 0.84%.

Luego se realizaron otros dos experimentos. En el segundo experimento se utilizaron las instancias mal clasificadas del primer experimento, siendo 582 instancias para entrenar la misma red GCNN y 600 epochs, obteniendo una tasa de error de 1.39%. Por último, el tercer experimento uso también las instancias mal clasificadas del segundo experimento, siendo en estos 1595 ejemplos y además 300 epochs, obteniendo una tasa de error de 1.31%. De este modo, el último experimento en el que se unió la salida de los tres anteriores obtuvo un valor de 0.65% de error.

5.1.3 Efficient Learning of Sparse Representations with an Energy-Based Model

El estudio descrito en [Ranzato2006], además del uso de CNNs, describe un modelo formado por tres componentes principales. Dicho modelo convierte los datos de entrada en una representación dispersa (sparse) con el objetivo de obtener un conjunto de datos más sencillo y compacto que tras ser introducido en la red neuronal convolucional genere características dispersas. En la Ilustración 26 se pude observar la estructura del sistema construido con dicho propósito y cuyos módulos tienen la siguiente función:

- El ***encoder**:* se trata de un conjunto de filtros que generan un vector partiendo de una imagen.
- El ***Sparsifying Logistic**:* modulo no-lineal que transforma el vector generado por el *encoder* en un vector disperso con valores entre [0,1]
- El ***decoder**:* conjunto de filtros que reconstruye la imagen de entrada partiendo del vector disperso.

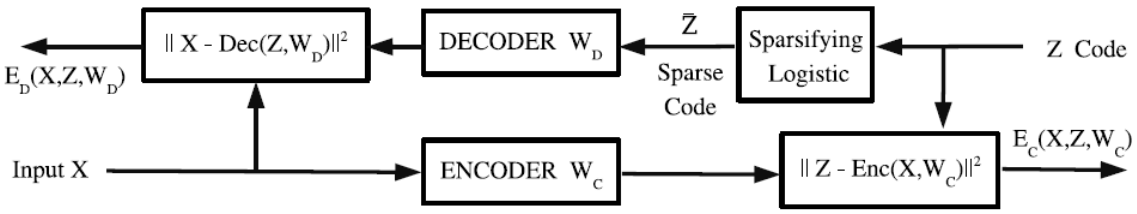

Ilustración 26: Estructura del sistema desarrollado

La arquitectura de la CNN utilizada en [Ranzato2006] es idéntica a la red LeNet-5 presentada en [Lecun1998] con la salvedad de utilizar un número mayor de filtros para la fase de extracción de características. De modo que si LeNet-5 utilizaba 6-16-120 filtros en las capas de convolución, en esta red se aumentó a 50-50-200 filtros.

Los experimentos realizados con esta red y el dataset MNIST original utilizan 55.000 instancias para la fase del entrenamiento y 5.000 para la validación, obteniendo una tasa de error del 0.7%. Otro de los experimentos que se llevó a cabo, fue la utilización de parches de

tamaño 5x5 obtenidos del conjunto de imágenes de entrenamiento originales para realizar el entrenamiento de este modelo, obteniendo un valor de 0.6% de tasa de error.

Por último, siguiendo las prácticas descritas en [Simard2003], se amplió el conjunto de entrenamiento con instancias obtenidas al realizar distorsión elástica en el conjunto original. Y se realizó un entrenamiento con el mismo modelo ya descrito, obteniendo una tasa de error de 0.39%.

De este modo, se puede observar que el hecho de introducir las pautas señaladas en [Simard2003] tuvieron una repercusión real en los resultados obtenidos, de modo que se considera un estudio muy interesante para su utilización en el presente trabajo y por eso se realizará una breve descripción de él a continuación.

5.1.4 Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis

Tras verificar que las CNNs son una tecnología muy potente para la tarea de clasificación de imágenes con contenido textual en [Lecun1998], se comenzaron a utilizar combinadas con una gran diversidad de métodos, que en algunos casos no mejoraban el rendimiento. Por eso, en el trabajo realizado en [Simard2003], se presenta un conjunto de buenas prácticas, con el objetivo de guiar el uso del modelo de redes neuronales convolucionales para obtener resultados buenos.

La práctica más importante hace hincapié en el conjunto de datos de entrenamiento, y es que, se debe de configurar un conjunto de datos cuyo tamaño sea el más grande posible, introduciendo si es necesario datos con distorsión para que realice una labor de aprendizaje más profunda. La segunda pauta fijada por [Simard2003] es la utilización de las redes convolucionales frente a redes no convolucionales, ya que éstas no requieren métodos complejos.

Para probar la utilidad de las prácticas a las que hace referencia, realizó experimentos basados en la creación de un nuevo conjunto de datos distorsionados y la utilización de las CNNs.

La distorsión simple de imágenes se basa en la aplicación de un desplazamiento a éstas. Es decir, para cada píxel se calcula una nueva ubicación de destino con respecto a la ubicación original de este. Esa diferencia entre el valor original y el calculado es la distorsión que se introduce en la imagen. El tipo de distorsión que mejores resultados de rendimiento ha generado en el dataset MNIST es la distorsión elástica. Para ello se utilizó un valor de desplazamiento Δx y Δy aleatorio entre [-1,1], a los que luego se aplicó filtros Gaussianos de desviación estándar (σ) y una normalización a un factor de escala (α). En la Ilustración 27 se

puede observar los resultados de su aplicación con distintos valores de coeficiente de elasticidad (σ) y de intensidad de deformación (α).

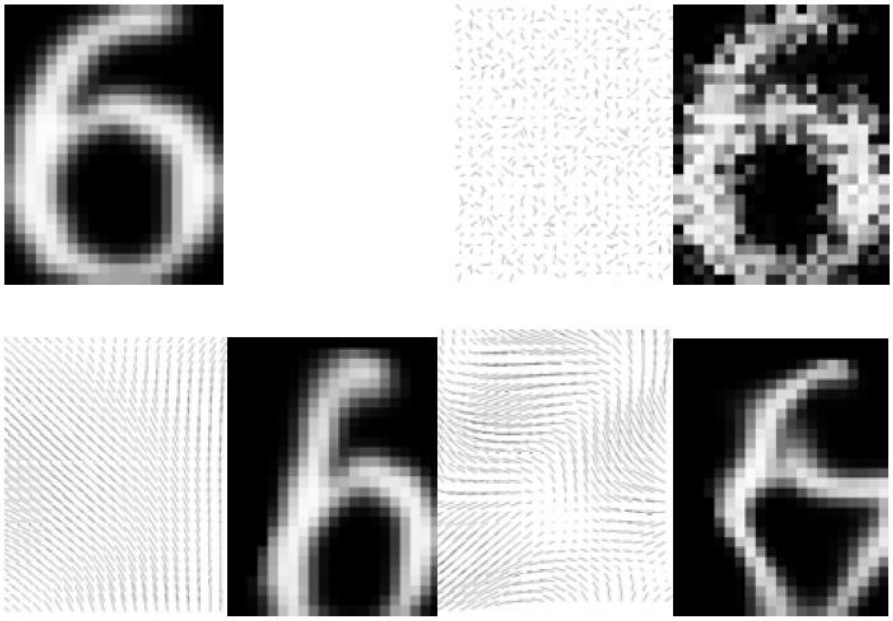

Ilustración 27: izda. superior: imagen original, dcha. superior: deformación, abajo: resultado aplicación

La arquitectura de CNN utilizada para certificar su buen rendimiento es la mostrada en la Ilustración 28. Dicha red está compuesta por dos capas de convolución, C1 que recibe como entrada a la red las imágenes distorsionadas con un tamaño 29x29 y en la que se aplica 5 filtros con un kernel de tamaño 5x5 y C2, con el mismo kernel y en la que se aplican 50 filtros. Intercaladas entre las capas de convolución se realizan las de submuestreo, en este caso se utiliza average pooling con un kernel de 2x2. Luego se aplican dos capas totalmente conectadas, la primera F3 con 100 neuronas y la segunda que además es la capa de salida con 10 neuronas.

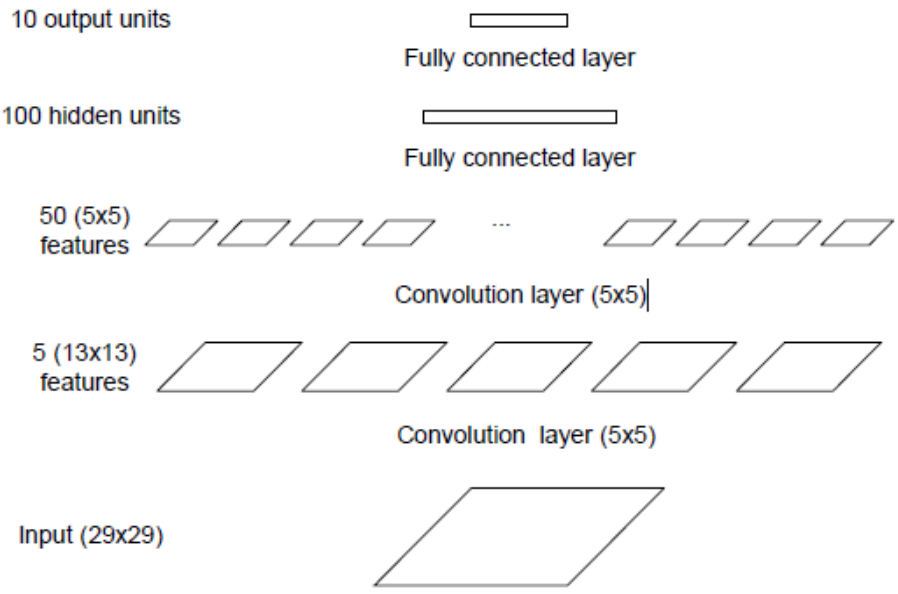

Ilustración 28: Arquitectura de CNN resultante tras introducir nuevas mejoras

De este modo, en los experimentos realizados en este estudio se utilizó el modelo de CNN descrito y el conjunto de datos MNIST formado por 50.000 instancias de entrenamiento a las que se aplicó distorsión elástica y 10.000 instancias de testing manteniendo su estado original. Es decir, no se agregaron instancias distorsionadas al conjunto de entrenamiento MNIST original, sino que a este conjunto se le aplicó la distorsión descrita, ya que se trata de un número de instancias adecuado para la tarea que se quiere resolver. Con dicho experimento se obtuvo una tasa de error del 0.4%, resultado que mejora el rendimiento de usar una red neuronal convolucional debido al uso de una etapa de procesamiento de las imágenes.

5.1.5 Unsupervised Learning of Invariant Feature Hierarchies with Applications to Objection

En el estudio [Ranzato2007] se propone un sistema innovador basado en un método de aprendizaje jerárquico no supervisado basado en la detección de características dispersas que son invariantes ante pequeños cambios o distorsiones.

El sistema utiliza una arquitectura de CNN combinada con un procedimiento de entrenamiento no supervisado a nivel de capas, reduciendo así, el problema de sobre parametrización de los procedimientos de aprendizaje supervisado, y obteniendo un rendimiento bueno con pocas instancias de entrenamiento etiquetadas.

De este modo, la arquitectura del sistema que proponen está compuesta por varios niveles. Cada uno de estos niveles está compuesto por dos capas, una primera capa de convolución en la que se aplican un banco de filtros a la entrada y la segunda capa de pooling, en la que se elige al valor máximo. La siguiente capa incluye su correspondiente función de activación,

en este caso una función sigmoide. Finalmente, se agrega una última capa entrenada en modo supervisado para realizar la clasificación, siendo este el elemento diferenciador, es decir, si se apilan niveles como los descritos se crea una arquitectura que esencialmente es idéntica a la de las CNNs. Las arquitecturas compuestas por numerosos filtros y características entrenables, requieren numerosos datos etiquetados, situación que puede no darse. Es por eso, que en este estudio, se pretende seguir la línea de estudios en los que se ha demostrado las ventajas que proporciona un pre-entrenamiento de las capas en modo no supervisado para obtener características invariantes que luego serán utilizadas para clasificar con entrenamientos supervisados con muy pocos datos.

Por esto, como la principal manera de obtener características invariantes es mediante la utilización de características dispersas, se ha combinado el sistema descrito anteriormente con la obtención de características dispersas construyendo una arquitectura de codificación-decodificación novedosa como se muestra en la Ilustración 29.

En esta arquitectura, se tiene una fase de codificación en la que se introduce las imágenes de tamaño 17x17 y se aplica un nivel como el descrito, en que se aplica una capa de convolución con 4 filtros y un kernel 7x7, una activación con una función sigmoide de no-linealidad y una capa de max pooling. Con estas capas se obtiene un vector de características invariantes que representa la imagen de entrada y los parámetros de transformación que indica la ubicación exacta de estos en la imagen. Este vector y estos parámetros de transformación son los que se utilizarán para realizar la decodificación en la etapa siguiente y en base a los que se realizará un algoritmo de aprendizaje supervisado para la última capa de clasificación.

Partiendo de este estudio, se realizó una experimentación con el dataset MNIST. Este consistió en la creación de una arquitectura compuesta por dos niveles. El primer nivel utilizando 5 filtros y un kernel de 7x7 para la capa de convolución, y un kernel de 2x2 para la capa de max pooling. Y un segundo nivel, con 1280 filtros y un kernel de 5x5, seguido de otro max pooling 2x2. Finalmente se agregaron dos capas totalmente conectadas, una usando 200 neuronas y otra final de 10 neuronas. Así, con esta red se realizó inicialmente un experimento con el enfoque de aprendizaje no supervisado y el dataset MNIST completo, obteniendo una tasa de error de 0.64%, para luego realizar una experimentación de aprendizaje supervisado utilizando las características obtenidas y diferentes tamaños de dataset MNIST. Por ejemplo, para 300 instancias de entrenamiento se obtuvo un resultado de 10.63% de tasa de error, pero para 60.000 instancias del dataset MNIST completo, se mejoró obteniendo 0.62% de tasa de error.

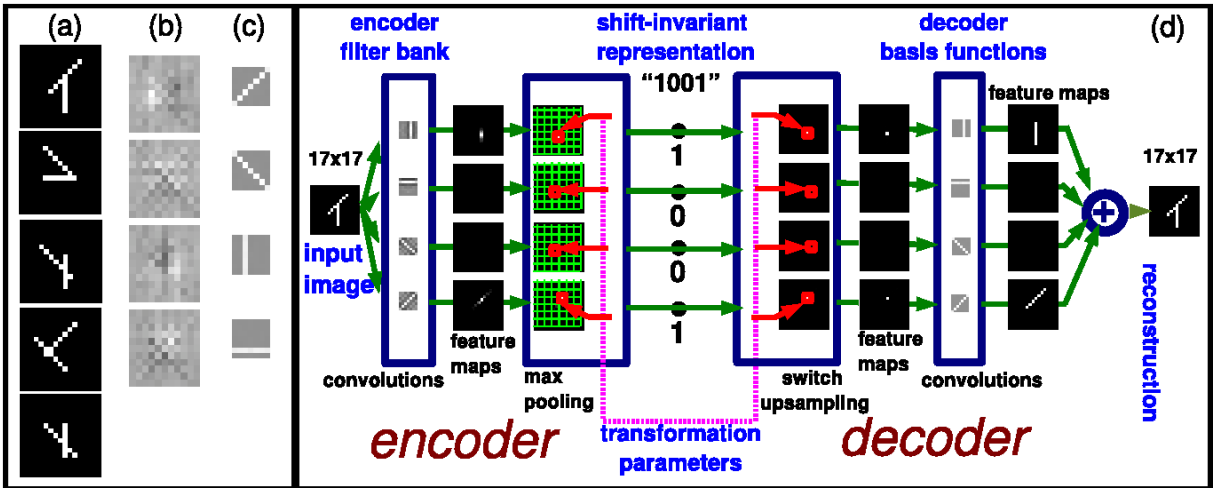

Ilustración 29: Extractor de características

5.1.6 Flexible, High Performance Convolutional Neural Networks for Image Classification

Gran parte de los estudios descritos hasta el momento fueron revisados en [Ciresan2010] cuyo autor posteriormente centraría sus estudios en la mejora de algunos métodos y prácticas utilizadas en las soluciones propuestas. De este modo, en [Ciresan2011A] se presenta una arquitectura de red neuronal en la que se fusionan varios conceptos de otros estudios, intentando mejorarlos.

Una de las cosas que destaca es la utilización del método simple de back-propagation que se utilizaba en las primeras CNNs, frente a la dinámica de los últimos estudios que se decantaban por el desarrollo de métodos innovadores para la extracción de características y la fase de aprendizaje.

Por lo tanto, [Ciresan2011A] propone una red basada en [Lecun1998] haciendo uso del método simple de back-propagration (Ilustración 30) para el entrenamiento de ésta y dando lugar a un proceso de aprendizaje sorprendentemente rápido. Otro de los aspectos que destaca es la utilización de la función max pooling en la capa de submuestreo en vez de utilizar la función average, esto se debe a que el max pooling puede converger de manera más rápida, ya que selecciona un mayor número de características invariantes que la función average y mejora la generalización.

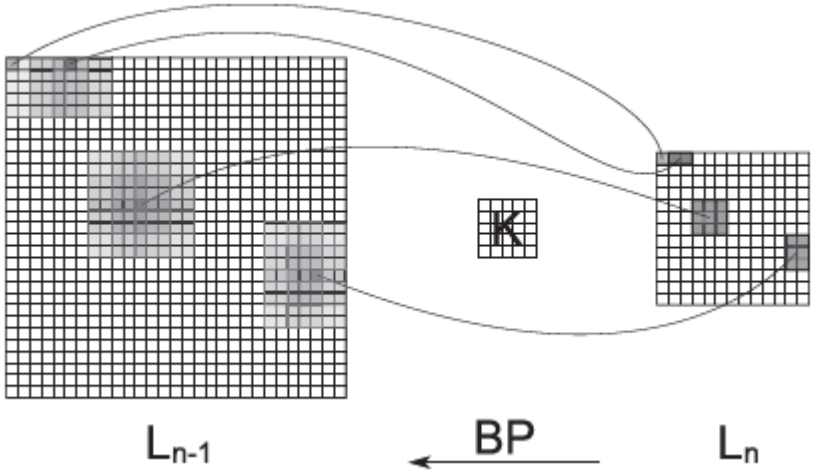

Ilustración 30: Back propagation

Además de todas estas decisiones, se realizó un plan de experimentación modificando la arquitectura de la red neuronal convolucional hasta obtener la red que mejoró los resultados del estado del arte. Dicha red está compuesta por 6 bloques conformados por una capa de convolución, una capa de activación y una capa de max pooling, utilizando en las capas de convolución 20-40-60-80-100-120 filtros respectivamente y un kernel de 5x5. En las capas de max pooling se utiliza un kernel de 2x2. La función utilizada en las capas de activación es una tangente hiperbólica escalada. Por último, se agregan las capas fully conected, una primera con 150 neuronas y para acabar una capa con un número de neuronas igual al número de clases. Otros de los parámetros que se determinaron fue el uso de un ratio de aprendizaje de 10^{-3}. Este experimento dio como resultado una tasa de error de 0.35%, siendo el mejor de los probados, no obstante, para llegar a él, se realizaron otras pruebas.

Uno de esos experimentos anteriores, es el realizado con un modelo red mucho más simple formado por 3 bloques iguales a los definidos, con la salvedad del número de filtros usados en las capas de convolución. En este caso se utilizaron 20-60-100 filtros, seguidos de una capa completamente conectada de 150 unidades, el resultado obtenido con esta red fue de 0.38% de tasa de error, siendo un valor realmente bueno si se tiene en cuenta que el modelo de red es bastante más simple que el que consigue 0.35%. En estos casos se debe de tener en cuenta si el uso de los recursos es un elemento limitador o no.

5.1.7 Convolutional Neural Network Committees For Handwritten Character Classification

Ahondando en el problema de reconocimiento de dígitos, Ciresan realizó una nueva investigación [Ciresan2011B]. En este estudio se realiza un proceso de experimentación más avanzado creando un comité o grupo de 7 redes neuronales convolucionales que se usarán para realizar la fase de aprendizaje. La arquitectura individual de red utilizada es idéntica y está formada por dos capas de convolución y max pooling intercaladas, siendo la primera

capa de convolución de 20 filtros cuyo tamaño de kernel es de 4x4 y la segunda de 40 filtros de 4x4. Para la primera capa de max pooling se utiliza un tamaño de kernel de 2x2 y para la segunda 3x3. Finalmente, se agregan dos capas completamente conectadas, la primera de 150 neuronas y la segunda con el número de clases a clasificar.

En el proceso seguido para entrenar el sistema, lo que si difiere son los conjuntos de datos utilizados para entrenar la red. Es decir, se ha partido de 7 conjuntos de datos, el dataset MNIST original y 6 MNIST preprocesados, en los cuales se han agregado distorsiones para agregar características diferentes. Una de las distorsiones utilizada es la más comúnmente utilizada, la distorsión elástica, con unos valores de $\sigma = 6$ y $\alpha = 36$. Tras esto, se realizó un proceso de normalización de todos los caracteres preprocesados a 10, 12, 14, 16, 18 y 20 pixeles. Luego se realiza el entrenamiento de cada dataset de manera individual.

Finalmente, el resultado del comité está formado por el promedio de las salidas individuales de cada uno de los entrenamientos. Cada uno de los dataset se entrena 5 veces con el mismo número de epochs, pero con la CNN inicializada de manera diferente, permitiendo realizar un análisis de error de una salida de $5^7 = 78125$ posibles comités de 7 redes cada una entrenada en uno de los 7 datasets. Los resultados obtenidos se representan con la media y la desviación estándar. El procedimiento descrito es el que se muestra en la Ilustración 31.

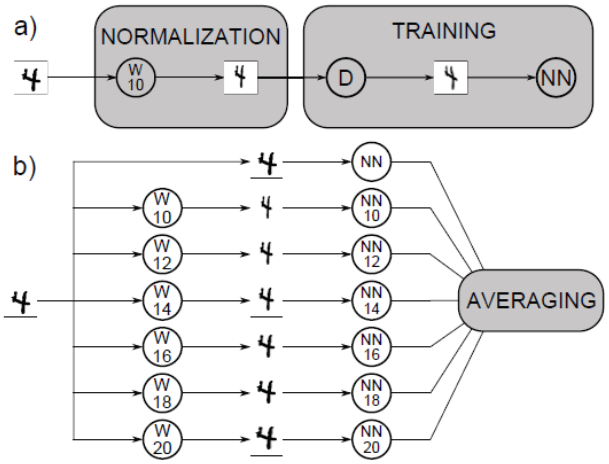

Ilustración 31: Procedimiento realizado a cada dataset y procedimiento final

La experimentación llevada a cabo se caracteriza por la utilización de un número de 800 epochs, obteniendo una tasa de error de 0.27% ± 0.02%.

5.1.8 Better digit recognition with a committee of simple Neural Nets

La continuación del estudio [Ciresan2011B] se realizó en [Meier2011], en el que se presentó un nuevo método para entrenar el comité agregándole una capa oculta a la red neuronal convolucional. El procedimiento seguido se basa en preprocesar los datos de entrenamiento para cada modelo en vez de entrenar varias redes con un subconjunto de datos de

entrenamiento, consiguiendo que el error no esté correlacionado. Esto se realiza para que los diferentes clasificadores se complementen, ya que los datos mal clasificados normalmente no se superponen, sino que ofrecen información que podría ser aprovechada por el comité al ser complementaria.

Por lo tanto, la estrategia seguida es la de producir un grupo de clasificadores consiguiendo que sus errores no estén correlacionados. Para ello, se usa el enfoque en el cual se procesan los datos de distinta forma, pero los clasificadores son idénticos usando la misma función de activación en el clasificador y así poder combinar los resultados.

Para MNIST se obtuvo una tasa de error de 0.39% usando un comité formado por 25 redes neuronales y una capa oculta.

5.1.9 Multi-column Deep Neural Networks for Image Classification

En el trabajo [Ciresan2012] se avanza un paso más en el estudio de las redes neuronales convolucionales, desembocando en redes neuronales convolucionales profundas conocidas como DNN (Deep convolutional neural networks). Se plantea el uso de las DNN para demostrar que la inicialización o el preentrenamiento no supervisado no es útil.

Las DNNs propuestas se caracterizan por tener los pesos inicializados de manera aleatoria, estar compuestas por varias capas (6-10) que incluyen no linealidad, no usar heurísticos adicionales, uso de la técnica de agrupación máxima en la que se selecciona la neurona más activa para las capas posteriores, composición de la parte superior de la jerarquía por perceptones multicapa estándares y combinación de columnas DNN creando una arquitectura MCDNN como la mostrada en la Ilustración 32.

La arquitectura de red DNN utilizada en los experimentos, está compuesta por dos bloques compuestos por una capa de convolución con 20 y 40 filtros respectivamente y un kernel de 4x4 y 5x5, las capas de submuestreo utilizan max pooling con un kernel de 2x2 y 3x3 respectivamente, seguidos de 2 capas completamente conectadas, la primera de 150 unidades y la última con tantas unidades como clases se quieran clasificar. Se utiliza una tangente hiperbólica escalada como función de activación entre capas de convolución y pooling y por último para clasificar una capa softmax. Un ejemplo de aplicación de las capas a una imagen del dataset se muestra en la Ilustración 33.

Para la experimentación se utilizó entre otros el dataset MNIST, al que se realizó distintas distorsiones para su uso en la fase de entrenamiento. De este modo, el plan de experimentación llevado a cabo fue, la realización de 5 experimentos para las imágenes con las distintas distorsiones realizadas y para el dataset original. Se obtuvo el peor resultado con los datos originales, con un valor de media de las 5 experimentaciones de 0.47% ± 0.05 de tasa de error y los mejores resultados con un valor de tasa de error de 0.39 ± 0.08. Estos experimentos posteriormente fueron combinados creando una arquitectura MCDNN con 35

experimentos, dando una tasa de error de 0.23%. Los resultados obtenidos por la red MCDNN se pueden ver en la Ilustración 34.

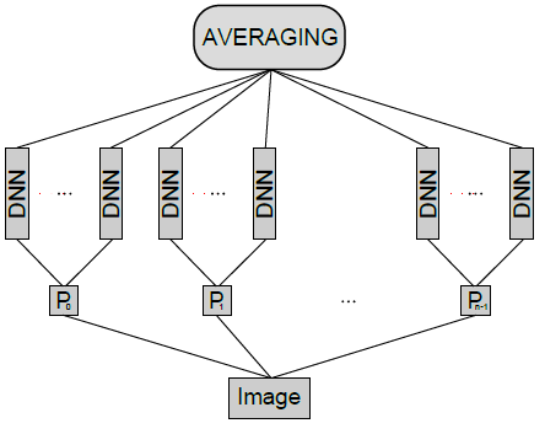

Ilustración 32: Arquitectura MCDNN

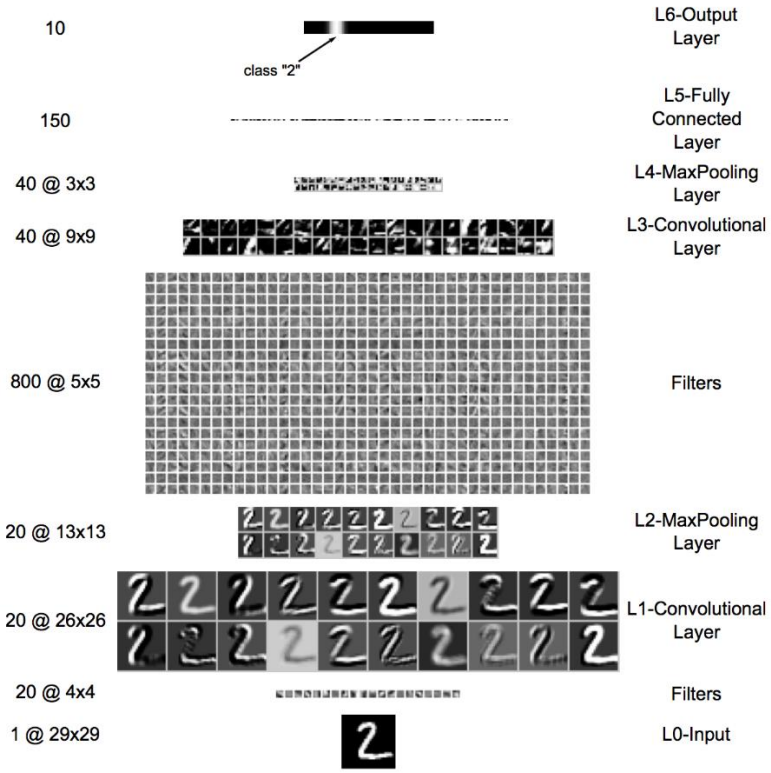

Ilustración 33: Aprendizaje que se realiza en la DNN paso a paso

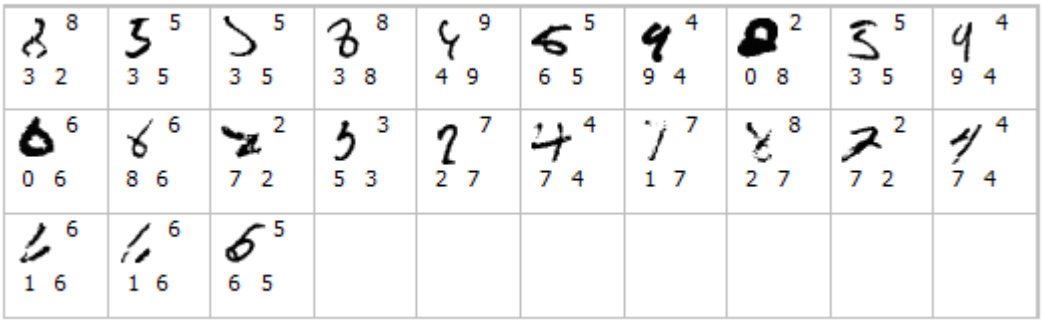

Ilustración 34: Errores obtenidos por la red MCDNN, en la esquina superior aparece la etiqueta real y abajo las predicciones

5.1.10 Beyond Human Recognition: A CNN-Based Framework for Handwritten Character Recognition

En el estudio [Chen2015] se busca mejorar los sistemas de reconocimiento de caracteres basados en CNN que existen hasta el momento, tratando de simplificar los pasos a seguir para crear y entrenar un modelo de CNN. Se pretende crear un sistema que se vea como una "caja negra" en la que se introduzcan los datos de entrada y se obtengan los resultados de la clasificación. Para ello proponen un framework compuesto por 3 partes como se puede ver en la Ilustración 35.

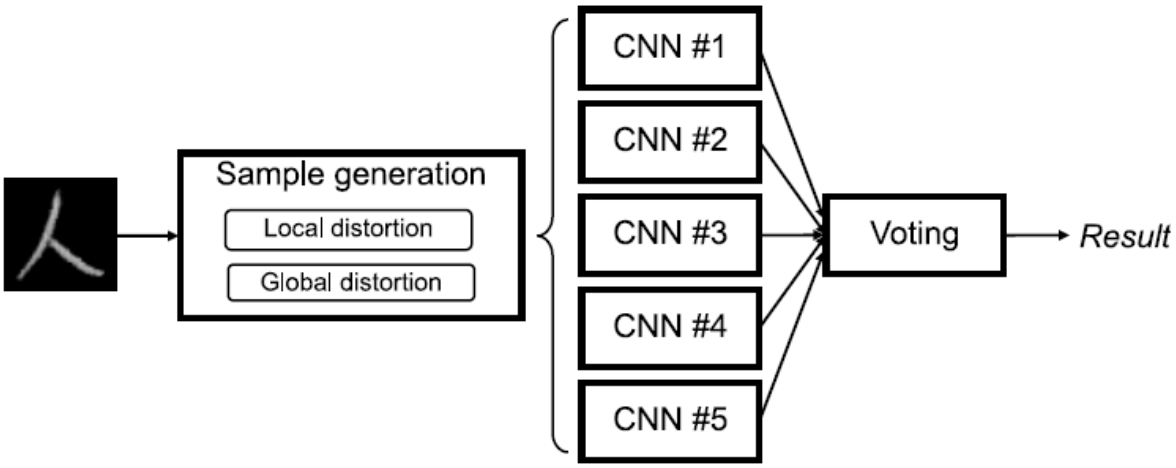

Ilustración 35: Framework propuesto por 3 partes

Como ya se ha comentado, el tamaño del conjunto de entrenamiento es uno de los factores más relevantes para obtener buenos resultados de la red, por eso, la primera parte se centra en la generación de un conjunto de datos de entrenamiento adecuado. Se utiliza dos tipos de distorsiones distintas para enriquecer el conjunto de entrenamiento, distorsión global y distorsión local.

La segunda parte se basa en la buena elección del modelo de CNN utilizado, siendo adecuado para el reconocimiento de caracteres y siguiendo las mejores prácticas.

Por último, se utiliza una fase de votación en la cual, partiendo de los modelos entrenados se usa la red cuyo resultado haya sido mejor. Este mecanismo puede mejorar significativamente el resultado.

De este modo, a la hora de realizar la experimentación con el dataset MNIST, inicialmente se realizó la generación de imágenes con distorsión local y distorsión general que fueron utilizadas como conjunto de entrenamiento. Luego se definió el modelo a entrenar, en este caso la arquitectura de dicho modelo está compuesta por tres bloques en los que se realiza una capa de convolución y otra de max pooling, tras los bloques se utiliza una capa completamente conectada y una capa de salida. Las características de las capas son: Conv1 con 32 filtros y un kernel de tamaño 5x5, MaxP2 con un kernel de 3x3, Conv3 con 64 filtros y un kernel de 3x3, MaxP4 de 3x3, Conv5 de 128 filtros y kernel de 3x3, MaxP6 con kernel de 3x3. Las capas de activación intercaladas utilizan una función ReLu, Full7 con 1200 neuronas y la capa de salida utiliza una función Softmax con tantas neuronas como clases.

De este modo, utilizando el framework propuesto se alcanzó un error de 0.18%, reduciendo el valor obtenido por la CNN propuesta que obtenía 0.23%.

5.2 Estado del arte del reconocimiento de caracteres basado en CNNs

Con el objetivo de ampliar la tarea de reconocimiento a un conjunto final formado por los caracteres alfanuméricos, también se tiene en cuenta aquellos estudios que hacen referencia a conjuntos de caracteres formados por letras, buscando de esta forma tener conocimiento de las posibles diferencias que puedan existir para crear un sistema que obtenga el mayor rendimiento posible. En el proceso de esta búsqueda se han encontrado varios estudios que resultan interesantes para mejorar el resultado del objetivo final.

5.2.1 End-to-End Text Recognition with Hybrid HMM Maxout Models

En [Alsharif2013]se propone una nueva solución para el reconocimiento de texto mediante un sistema de reconocimiento de extremo a extremo. Creando un sistema en el que se va realizando la tarea de reconocimiento de menor a mayor dificultad, es decir, el reconocimiento de texto se basa en el subproblema de reconocimiento de palabras, que a su vez se basa en el subproblema de reconocimiento de caracteres, por lo que este sistema se apoya en el resultado de uno para conseguir el siguiente nivel.

El sistema planteado combina las redes Maxout [Goodfellow2013] y los modelos híbridos HMM. No obstante, desde el punto de vista de este trabajo, el problema que se pretende

resolver es el de reconocer caracteres, por lo que se centrará la descripción del trabajo realizado para llevar a cabo el reconocimiento de caracteres individuales.

Para el reconocimiento de caracteres se ha usado una CNN compuesta por cinco niveles, integrando en uno de ellos una red Maxout, siendo ésta una capa completamente conectada que utiliza dropout para regularizar la red. De este modo, los tres primeros niveles estan constituidos por un conjunto de capas de convolución y pooling, seguidos de un cuarto nivel formado por una capa de Maxout y finalmente una capa completamente conectada con una función de salida Softmax. Las tres primeras capas de convolución usan 48, 128 y 128 filtros respectivamente, utilizando las dos primeras un kernel de tamaño 8x8 y la tercera un kernel de 5x5. Seguida de cada una de las capas de convolución va intercalada una capa de pooling, utilizando las dos primeras un kernel de 4x4 y la tercera uno de 2x2. La cuarta capa utiliza 400 neuronas y finalmente se agrega una capa totalmente conectada que utiliza una función Softmax con 62 neuronas.

La experimentación llevada a cabo para el reconocimiento de caracteres se basó en un experimento con el dataset ICDAR 2003, al que se aplicó un procesamiento para convertirlo a la escala de grises con un tamaño de 32x32. Los resultados de accuracy obtenidos fueron del 85.5 %.

5.2.2 Deep Features for Text Spotting

En [Jaderberg2014] se realiza un estudio completo proponiendo una solución al reconocimiento de texto, mejorando el estado del arte. El estudio aporta tres contribuciones novedosas. Una de estas contribuciones, es el desarrollo de una CNN para la tarea de detección y clasificación de imágenes naturales con texto. En el estudio, definen la tarea de reconocer caracteres como el núcleo del sistema, asignándole la importancia necesaria.

De este modo, para clasificar imágenes con un posible carácter o solo fondo, se extraen un conjunto de características de dichas imágenes, con las que luego un clasificador binario aprende por cada letra del alfabeto. Los clasificadores aprenden a producir una distribución de probabilidad sobre los caracteres que luego se maximiza para reconocer el carácter contenido en la imagen. El método de aprendizaje y clasificación utilizado en el sistema utiliza una CNN. Dicha CNN, está compuesta por bloques, compuestos a su vez por capas de convolución, capas de activación ReLu, capas de normalización y capa de max pooling. La optimización se lleva a cabo mediante la función SGD y se utiliza la técnica de back-propagation.

El proceso de aprendizaje para realizar la clasificación de caracteres se divide en dos etapas. En la primera etapa se realiza el aprendizaje de caracteres sin hacer diferencia entre mayúsculas y minúsculas a modo de aprendizaje base. Y en la segunda etapa, partiendo del resultado de la anterior etapa se aplica otro clasificador para obtener además de la

clasificación inicial, una clasificación entre imágenes con caracteres e imágenes sin caracteres, clasificación para distinguir las letras mayúsculas y minúsculas y un clasificador de bigrams.

Para este trabajo solo tiene relevancia la primera etapa del aprendizaje para el caso de caracteres sin diferenciar mayúsculas y minúsculas, y por lo tanto, será la única descrita.

Etapa 1: clasificador insensible a mayúsculas y minúsculas. Se trata de una CNN compuesta por cuatro capas que tiene como salida la probabilidad de pertenecer al alfabeto (26 letras), al conjunto de dígitos (10 dígitos) o a una clase que representa el caso en que no exista ningún carácter en la imagen, sino que solo sea fondo. Clasificando en total 37 clases diferentes.

Esta CNN recibe como entrada una imagen de tamaño 24x24 en escala de grises que contiene un único carácter, con el carácter centrado y normalizado. En la primera capa de convolución se le aplica a la imagen de entrada 96 filtros con un tamaño de kernel de 9x9, a continuación, se aplica una capa de max pooling con un tamaño de 2x2. La secuencia de capas continua con tres capas de convolución en las que se aplican 128, 512 y 148 filtros con kernels de tamaño 9x9, 8x8 y 1x1 respectivamente. Finalmente se agrega una capa completamente conectada para la clasificación utilizando 37 neuronas. El entrenamiento se realiza usando SGD y back-propagation, además se agrega una capa de dropout en todas las capas excepto en la primera capa de convolución, la proporción de dropout utilizada es 1, 0.5, 0.5, 0.5. Además, al dataset utilizado ICDAR 2003 se le ha agregado distorsión. Así, el resultado obtenido con esta arquitectura de CNN es de 91% de accuracy.

5.2.3 Image character recognition using Deep convolutional neural network learned from different languages.

El estudio [Bai2014] utiliza el dataset de caracteres ICDAR 2003 [Lucas2003]. En él se describen dos tipos de CNN, por un lado, se propone un tipo de red denominado shared-hidden-layer convolutional neural network (SHL-CNN) que consta de una red neuronal convolucional que usa capas ocultas. Estas capas ocultas son compartidas entre los caracteres en diferentes tareas siendo consideradas como un proceso de extracción de características universal para el entrenamiento basado en caracteres de múltiples idiomas, además, la última capa de softmax es dependiente de dichas tareas. Por otro lado, utiliza una red convolucional tradicional en la que utiliza los métodos convencionales y a la que denomina Con-CNN.

Los datos de entrada a la red son imágenes de tamaño 48x48. En la fase de experimentación, se utilizaron los 3 modelos de red diferentes para cada tipo descrito anteriormente, estando compuestos como se describe a continuación:

- **Modelo 1**: 64C5-MP2-CN-64C5-CN-MP2-64L3-32L3-62N

- **Modelo 2**: 64C7-MP2-CN-64C7-CN-MP2-64L5-32L5-62N
- **Modelo 3**: 64C9-MP2-CN-64C9-CN-MP2-64L7-32L7-62N

Los modelos presentados utilizan una nomenclatura para abreviar la descripción. Esta nomenclatura se interpreta de modo que el valor numérico anterior a la letra se corresponde con el número de filtros utilizados para dicha capa, seguido de las letras que representan el tipo de capa, siendo C una capa de convolución, MP una capa max pooling, CN una capa de normalización de contraste local, L una capa de convolución que no comparte los pesos y N capa completamente conectada. El número que sigue a la letra es el tamaño del kernel utilizado para ese tipo de capa. Además, se utiliza como capa de activación entre la capa de convolución y la de pooling una función ReLu.

Los errores obtenidos para el dataset ICDAR 2003 utilizando el tipo de red Con-CNN con los modelos de red presentados son de 0.156% para el modelo 1, 0.14% para el modelo 2 y 0.12% para el modelo 3.

5.2.4 Scene text recognition with CNN classifier and WFST-based word labeling

En el estudio [Liu2016] se describe un sistema para resolver el reconocimiento de texto en imágenes de escenas reales, problema que generalmente requiere como mínimo una división del problema real en dos subproblemas menores. Por una parte, una fase de detección de texto con su respectivo método de segmentación y por otra una fase de reconocimiento del texto que aparece en la zona recortada.

No obstante, como se ha comentado en los apartados anteriores, la tarea de detección y segmentación queda fuera de los objetivos de este trabajo, por lo tanto solo se describirá el método utilizado en [Liu2016] para la tarea de reconocimiento de caracteres. En concreto, el trabajo describe la utilización de un modelo CNN para el reconocimiento de caracteres de escenas reales utilizando para la experimentación el dataset ICDAR 2003.

De este modo, en la propuesta [Liu2016], se inicia la secuencia de procedimientos para reconocer texto con una CNN para la clasificación de caracteres. La arquitectura de CNN utilizada es la denominada Fractional Max-Pooling (FMP) [Wang2011], que utiliza características de aprendizaje supervisado. Dicha arquitectura, mostrada en la Ilustración 36 se compone de l bloques FMP. Cada bloque está compuesto por dos capas de convolución y una capa FMP, las capas de convolución utilizan un número de filtros dependiente del número de bloque al que pertenecen y kernels de 2x2 y 1x1 respectivamente. Así por ejemplo para el primer bloque de los doce utilizados para la experimentación se utilizarían $4N_{l-1}$ filtros para la primera capa de convolución y N_l para la segunda, siendo $N_l = 32l$ y siendo l el número de bloques FMP utilizados, además, la capa de convolución que utiliza un kernel de 1x1 es utilizada para agregar no linealidad al modelo. Así, finalmente tras apilar los l

bloques se agrega dos capas de convolución idénticas a las definidas en los bloques, seguidas de una capa completamente conectada en la que se utilizan 62 neuronas y la función de salida Softmax para realizar la tarea de clasificación. La función de activación elegida es ReLu. Se utilizan 62 neuronas ya que el conjunto que quiere clasificar está compuesto por 62 clases (26 letras en minúscula, 26 letras en mayúscula y 10 dígitos).

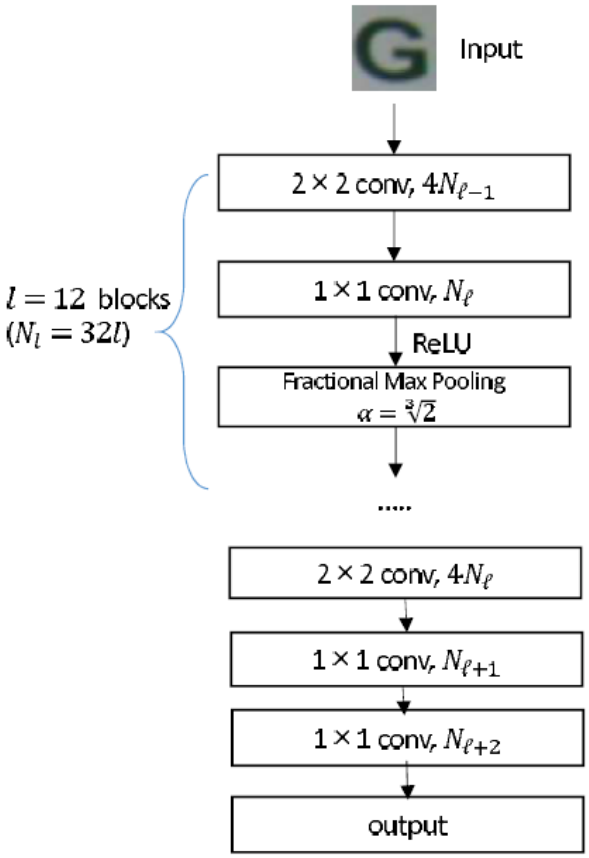

Ilustración 36: Arquitectura de la red FMP

El proceso de experimentación realizado en el estudio fue muy sencillo. Por una parte, se obtuvo para un único experimento de la red con el dataset ICDAR 2003 un resultado de 90.5% de accuracy, mientras que para un conjunto de 12 experimentos con el mismo dataset se obtuvo un resultado de 92.2% de accuracy.

5.3 Conclusiones

Como se puede observar en la Tabla 10, resumen de todas las arquitecturas descritas en el estudio teórico para el dataset MNIST, existe una gran diversidad de modelos de redes neuronales convolucionales que proporcionan resultados muy satisfactorios para la tarea de reconocimiento de caracteres (mayoritariamente dígitos). Como es obvio, no se puede probar con todas las arquitecturas descritas, sino que habrá que realizar una elección de las redes cuyas características sean las más convenientes para las imágenes a tratar.

Es por este motivo, que se ha llegado a la conclusión de que para elegir el modelo que obtenga el mayor rendimiento y mejor resultado de clasificación, sería un buen punto de partida comenzar por la arquitectura de red descrita en el estudio pionero realizado en [Lecun1998]. Ya que en la gran mayoría de los trabajos que se han realizado posteriormente y que han obtenido una mejora de los resultados, han tenido influencia de dicho trabajo. De este modo, en el actual proyecto se seguirá la esta misma línea, partiendo de la red propuesta en [Lecun1998] se irán realizando modificaciones en base a estudios posteriores. Las modificaciones por realizar se centrarán sobre todo en la utilización de diferentes funciones de activación y optimización y el número de capas de convolución y pooling, además de variar el número de filtros utilizado en éstas. De modo que se descartarán aquellas redes que utilicen en sus arquitecturas parámetros que no aporten mejoras. Además, cabe destacar que el dataset utilizado en este caso no está compuesto únicamente de dígitos por lo tanto también se tendrá en cuenta las redes estudiadas para un conjunto ampliado con letras. Un resumen de los estudios encontrados en este caso son las arquitecturas descritas en la Tabla 11, estudios que han utilizado como dataset de referencia a ICDAR 2003. Algunos de estos estudios también serán utilizados para la fase de experimentación.

Arquitecturas	*Modelo de red*	*Función de activación*	*Función de salida*	*Optimizador*	*Distorsión*	*Tasa Error*
LeNet-5 *[Lecun1998]*	6C5-AP2-16C5-AP2-120C5-F84-10N	Sigmoide	Tangente Hiperbólica	SGD	No	**0.95%**
					Si	**0.8%**
LeNet-4 *[Lecun1998]*	4C5-AP2-16C5-AP2-F84-10N				No	**1.1%**
Boosted LeNet-4 *[Lecun1998]*					No	**0.7%**
GCNN *[Calderon2003]*	12G (2Freq y 6Ori)-AP2-16C5-AP2-120C5-F84-10N	Sigmoide	Tangente Hiperbólica	SGD	No	**0.84%**
LeNet-5-Mod *[Ranzato2006]*	50C5-AP2-50C5-AP2-200C5-F84-10N	Sigmoide	Tangente Hiperbólica	SGD	No	**0.7%**
					Si	**0.39%**
[Simard2003]	5C5-AP2-50C5-AP2-F100-10N	Sigmoide	Tangente Hiperbólica		Si	**0.4%**
[Ranzato2007] *No supervisado*	5C7-MP2-1280C5-MP2-F200-10N	Sigmoide	Softmax	SGD	No	**0.64%**
[Ciresan2011A]	20C5-MP2-40C5-MP2-60C5-MP2-80C5-MP2-100C5-MP2-120C5-MP2-F150-10N	Tangente Hiperbólica	Softmax	-	No	**0.35%**
[Ciresan2011A]	20C5-MP2-60C5-MP2-100C5-MP2-F150-10N				No	**0.38%**
[Ciresan2011B]	Comité 7 (20C4-MP2-40C4-MP3-F150-10N)	Tangente Hiperbólica	Softmax	-	Si	**0.27% ± 0.02%**
[Meier2011]	Comité 25 (20C4-MP2-40C4-MP3-F150-10N)	ReLu	Softmax	-	Si	**0.39%**
[Ciresan2012]	20C4-MP2-40C4-MP3-F150-10N	Tangente hiperbólica	Softmax	-	No	**0.47% ± 0.05**
	MCDNN (20C4-MP2-40C4-MP3-F150-10N)				Si	**0.23%**
[Chen2015]	32C5-MP3-64C3-MP3-128C3-MP3-F1200-10N	ReLu	Softmax	-	Si	**0.23%**

Tabla 10: Resumen estado del arte de modelos de CNNs para el dataset MNIST

Como se ha comentado, también se tendrá en cuenta los modelo de CNNs encontrados para el dataset ICDAR 2003 en el que se utiliza un conjunto de caracteres alfanuméricos. En este caso los modelos, características y tasa de error obtenidos son los mostrados en la Tabla 11.

Arquitecturas	*Modelo de red*	*Función de activación*	*Función de salida*	*Distorsión*	*Accuracy*	*Tasa Error*
[Alsharif2013]	48C8-MP4-128C8-MP4-128C5-MP4-MAX400-62N	ReLu	Softmax	No	**85.5%**	-
[Jaderberg2014]	96C9-MP2-128C9-D1-MP2-512C8-D0.5-MP2-148C1-D0.5-MP2-D0.5-37N	ReLu	Softmax	Si	**91%**	-
Con-CNN [Bai2014]	64C5-MP2-CN-64C5-CN-MP2-64L3-32L3-62N	ReLu	Softmax	Si	-	**0.156%**
	64C7-MP2-CN-64C7-CN-MP2-64L5-32L5-62N				-	**0.14%**
	64C9-MP2-CN-64C9-CN-MP2-64L7-32L7-62N				-	**0.12%**
[Liu2016]	12 bloques(1408C2-384C1-FMP) + 1536C2-416C1-448C1-62N	ReLu	Softmax	Si	**92.2%**	-

Tabla 11: Resumen estado del arte de CNNs para dataset ICDAR 2003

6. Conjuntos de datos

En este trabajo se han utilizado dos conjuntos de datos diferentes para llevar a cabo el plan de experimentación. Inicialmente se ha experimentado con un conjunto de datos compuesto por un número de caracteres más reducido, tratándose de imágenes en las que aparecen los dígitos del 0-9, imágenes donde no hay caracteres e imágenes donde aparece una sola letra, la A. Luego se amplió introduciendo imágenes con letras nuevas, siendo el objetivo obtener un conjunto formado por todos los caracteres alfanuméricos. Otro aspecto en el cual difieren los conjuntos es en el preprocesamiento aplicado a las imágenes, mientras que el primer conjunto no tiene ningún tipo de procesamiento, el segundo sí.

6.1 Conjunto inicial (CJ1)

El conjunto inicia está compuesto por las imágenes resultantes del proceso de segmentación descrito en el apartado de sistema actual. Es decir, se trata de las imágenes obtenidas de la captura de las cámaras sin ningún tipo de procesamiento, realizándole exclusivamente la segmentación de cada dígito en una nueva imagen.

El conjunto está compuesto por imágenes que contienen dígitos, fondo y la letra A, siendo en total 12 clases que se asignan del siguiente modo:

- **Clases de 0-9 (C0, C1, C2, …, C9**): la etiqueta que se asigna se corresponde con el dígito que contiene la imagen.
- **Clase 10 (C10):** la etiqueta se corresponde con aquellas imágenes en las que no aparece ningún carácter, solo es fondo.
- **Clase 11 (C11):** la etiqueta se corresponde con la letra A.

Una muestra de las imágenes que componen el conjunto de datos es idéntica a la mostrada en la Ilustración 22.

La organización del dataset se trata de una estructura jerárquica de directorios como se puede ver en la Ilustración 37, de modo que de un directorio base cuelgan tantos directorios como clases tiene el dataset, que a su vez, contienen las imágenes que pertenecen a dicha clase, por ejemplo, en el directorio de la clase 0 se almacenan las imágenes que contienen el dígito 0 y así para todas las demás clases.

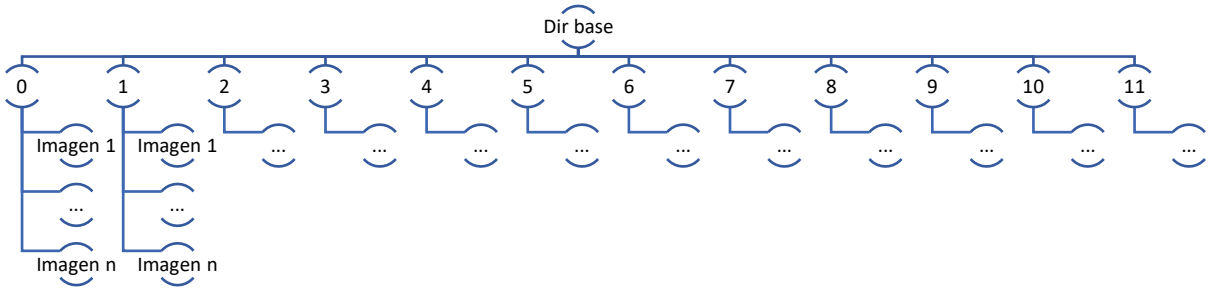

Ilustración 37: Jerarquía de directorios del dataset inicial

Así, la ruta de una imagen es: dirbase/clase/clase prob tiempo.tif

Cada una de las imágenes está nombrada siguiendo la nomenclatura: clase prob tiempo, siendo "clase" su etiqueta, "prob" la probabilidad con la que se clasifico por el script de Halcon y "tiempo", el valor en segundos del instante de tiempo en la cual fue capturada la imagen.

De este modo, siguiendo una proporción real de la aparición de cada carácter en las imágenes, se ha obtenido un conjunto de datos con el siguiente número de instancias de cada clase:

Clase	*Número de instancias*	*Porcentaje*
0	1.047	9%
1	1.467	13%
2	1.408	12%
3	990	9%
4	718	6%
5	80	1%
6	885	8%
7	332	3%
8	1.396	12%
9	436	4%
10	1.807	16%
11	712	6%
Total	**11.278**	

Tabla 12: Proporción de instancias de cada clase de CJ1

Creando de este modo un conjunto formado por un total de 11.278 instancias, siendo la proporción de cada clase la descrita en la Tabla 12.

Generalmente la división del conjunto global en subconjuntos de entrenamiento, testing y validación dependerá del experimento que se quiera realizar, pero de forma general este conjunto se ha divido inicialmente en una proporción 75-25, la proporción del 75% será utilizada para realizar el entrenamiento y la validación y el 25% restante para el testing. Además, el 75% se vuelve a dividir en una proporción 85-15, utilizando el 85% para el entrenamiento y el 15% restante para la validación. De este modo, se utilizan 8.461 instancias para realizar el entrenamiento y la validación, de las cuales 7.189 son utilizadas para el entrenamiento y 1.269 para la validación. El testing se realiza con 2.820 instancias.

6.2 Conjunto ampliado y preprocesado (CJ2)

Este conjunto se compone de imágenes iguales a las anteriores, es decir, cada imagen está recortada y contiene un único carácter, pero en este caso el conjunto de caracteres se ha ampliado agregando la aparición del conjunto letras {B, C, P, S, T, U, V, W, Y, Z} y además se le ha aplicado un proceso de manera previa, donde se lleva a cabo el cálculo de profundidad del carácter, resultando de este proceso una imagen en escala de grises y no a color como las originales.

Al ser agregadas al conjunto de caracteres nuevas letras, se ha tenido que etiquetar dichas letras para poder realizar su clasificación. Para ello se ha seguido la misma organización adoptada en el conjunto inicial, ampliándolo del siguiente modo:

- **Clases de 0-9 (C0, C1, C2, …, C9):** la etiqueta que se asigna se corresponde con el dígito que contiene la imagen.
- **Clase 10 (C10):** la etiqueta se corresponde con aquellas imágenes en las que no aparece ningún carácter, solo es fondo.
- **Clase 11-36 (C11, C12, …, C36):** la etiqueta se corresponde con la letra del abecedario que le corresponda a esa posición, es decir, comenzando en la posición 11, la primera letra del abecedario se ha asignado a la clase 11, las siguientes letras tienen asignada la clase que les corresponde siguiendo dicha secuencia. Así, la letra B tiene asignada la clase 12, la C la clase 13, la D la clase 14, hasta la Z que se corresponde con la clase 37. Otra salvedad del conjunto es que realmente no se tienen todas las letras del abecedario, sino solo algunas, por lo tanto, algunas clases aun no existen, pero se les ha asignado la clase para el caso de que aparezcan.

Una muestra del conjunto de imágenes que compone CJ2 es el mostrado en la Ilustración 38.

Ilustración 38: Imágenes que conforman el CJ2

La estructura de almacenamiento es idéntica a la del conjunto anterior, con la salvedad del número de clases. Por lo tanto, la ruta de una imagen es: dirbase/clase/id-clase-nom1-nom2.tif.

En este caso lo único que importa de la nomenclatura del identificador de cada imagen es que se utiliza el carácter guion ('-') como separador y que en la segunda posición se encuentra la clase a la que corresponde. Esto se debe a que inicialmente las imágenes se facilitaron en una sucesión de directorio (nom1, nom2 y clase) y solo tenían como identificador un id único, por lo tanto, se tuvo que realizar un re-etiquetado de las imágenes y se creó dicha nomenclatura en la cual lo importante es el carácter de separación y la segunda posición.

La proporción de instancias de cada clase en este conjunto ya no es la real, sino que se ha pretendido equilibrar las clases creando un conjunto balanceado. Objetivo que no se ha cumplido en el caso de las imágenes que contienen letras, ya que su aparición es mucho menos frecuente y no se ha podido realizar. Por lo tanto, el número de instancias de cada clase es la siguiente:

Clase	*Carácter*	*Número de instancias*	*Porcentaje*
0	0	1.044	7%
1	1	1.044	7%
2	2	1.044	7%
3	3	1.044	7%
4	4	1.044	7%
5	5	1.044	7%
6	6	1.044	7%
7	7	1.044	7%
8	8	1.044	7%
9	9	1.044	7%
10	No carácter	1.044	7%
11	A	1.117	8%
12	B	148	1%
13	C	140	1%
26	P	100	1%
29	S	480	3%
30	T	276	2%
31	U	132	1%
32	V	100	1%
33	W	100	1%
35	Y	100	1%
36	Z	108	1%
Total		**14.285**	

Tabla 13: Proporción de instancias de cada clase de CJ2

Así, la cantidad de instancias de cada clase de la que se compone el conjunto está descrita en la Tabla 13, teniendo un conjunto global formado por 14.285 instancias.

Al igual que con el conjunto anterior se realiza una división inicial de proporción 75-15 entrenamiento y validación, y testing. Luego, se divide el entrenamiento y la validación en una proporción 85-15. Por lo tanto, el subconjunto de entrenamiento está compuesto por 9.106 instancias, el de validación por 1.607 instancias y el de testing compuesto por 3.572.

7. Tecnologías y herramientas de desarrollo

Para llevar a cabo el desarrollo práctico del trabajo se han utilizado tecnologías y herramientas actuales cuyas características hacen que sean muy atractivas para tareas de Deep Learning y análisis de datos. Dichas tecnologías y herramientas son:

7.1 Python

Lenguaje de programación elegido para llevar a cabo el desarrollo. Es un software de código abierto creado por Guido Van Rossum a principios de los años 90. En los últimos años ha ganado gran popularidad y se ha generalizado su utilización para tareas de análisis de datos.

Las características que han fomentado dicha evolución creando un valor diferencial respecto a otros lenguajes de programación son:

- **Generalidad:** Puede utilizarse como lenguaje de programación para diferentes tipos de programas. Desde ser utilizado como lenguaje de scripting sustituyendo a otros lenguajes más limitados como AWK. Lenguaje para desarrollo web, principalmente gracias a frameworks de desarrollo web como Django. Y en los últimos años, el gran motor de evolución, que ha generado la extensión de Python es la gran utilización de la inteligencia artificial, el análisis de datos, etc generando un nuevo campo de trabajo en el que se utiliza Python como lenguaje.
- **Multiplataforma:** Existen versiones de Python para la gran mayoría de sistemas operativos, siendo únicamente necesario el intérprete correspondiente.
- **Interpretado:** A diferencia de otros lenguajes como Java o C, el código no se compila antes de su ejecución, sino que es interpretado en tiempo de ejecución. También es un lenguaje de tipado dinámico**.**
- **Interactivo:** Tiene un intérprete por líneas comandos donde poder probar líneas de código, ejecutarlas y obtener un resultado, pudiendo así comprender mejor el funcionamiento del lenguaje.
- **Multiparadigma:** Combina diferentes paradigmas de programación, aunque principalmente es orientado a objetos, pero también incorpora pequeñas nociones de programación imperativa, funcional, procedural y reflexiva.
- **Funciones y librerías:** Cuenta con muchas funciones ya incorporadas en el lenguaje y se pueden importar infinidad de librerías para tratar temas específicos como el Deep learning.
- **Sintaxis sencilla y clara**: Uno de los principios clave es la legibilidad del código, usando para ellos una notación identada como separación de porciones de código.

Por lo tanto, debido a la gran revolución y extensión de Python en el campo de la ciencia de los datos se ha utilizado para el presente trabajo, complementándolo como bien se señala en las características nombradas, con librerías y tecnologías para ciertas tareas específicas. Estas tecnologías y librerías serán descritas a continuación.

7.2 TensorFlow

Biblioteca de código abierto creada por Google para desarrollar sistemas de aprendizaje automático, siendo de las más utilizadas en problemas de Deep Learning. Utiliza gráficos de flujo de datos, donde los nodos representan las operaciones matemáticas, mientras que los bordes de dichas gráficas representan las matrices de datos multidimensionales comunicadas entre ellos. De este modo, se pueden construir redes neuronales que permiten detectar las relaciones y patrones existentes siguiendo el esquema realizado por un ser humano.

En la actualidad se utiliza de manera generalizada para el desarrollo de sistemas basados en redes neuronales convolucionales, por eso ha sido la elección elegida en este trabajo.

De manera específica, en el desarrollo de este sistema, se ha utilizado para procesar las imágenes de los conjuntos de datos, convirtiendo éstas en “tensores”, principal estructura de datos que se maneja en TensorFlow. De ese modo, al representar la información del problema en tensores, se le aplicarán algoritmos que transformarán estos datos aplicando diferentes operaciones y transformaciones, más específicamente las diferentes capas que conforman una CNN.

Para simplificar la creación del sistema y el manejo de las CNN, se utilizó otra de las bibliotecas del ecosistema denominada Keras.

7.3 Keras

Es una biblioteca de código abierta escrita en Python, que puede ejecutarse sobre TensorFlow, con el objetivo de agilizar el proceso de aprendizaje sobre CNNs. Keras permite realizar prototipos de manera fácil y rápida, crear CNNs y RNNs ejecutándolas sobre CPUs y GPUs.

En el sistema implementado se utilizado Keras para la construcción de las arquitecturas de CNNs a probar, así como realizar todo el proceso de compilación, entrenamiento y testing de dichos los modelos, generando como resultado las métricas de la evaluación de las arquitecturas, así como los pesos finales de las redes para su posterior utilización.

7.4 Otras bibliotecas, librerías y módulos utilizados

Además de las librerías utilizadas para la construcción, entrenamiento y evaluación de las CNNs, se han utilizado diversas librerías y herramientas para llevar a cabo la implementación del sistema. Las más importantes y cuyo uso es necesario se describen a continuación:

7.4.1 OpenCV

Biblioteca de código abierto para tareas de visión artificial desarrollada por Intel. Se ha utilizado para la carga de las imágenes del sistema, realizando el redimensionado de estas a un tamaño fijo para su posterior tratamiento.

7.4.2 Pillow

Se trata de la bifurcación o ampliación de la librería para edición de imágenes facilitada por Python (PIL, Python Imaging Library) que a partir de la versión 2.7 de Python dejo de ser desarrollada. Pillow se ha utilizado de manera conjunta con OpenCV para manejar los conjuntos de imágenes del problema. Soporta gran variedad de formatos de imágenes.

7.4.3 Numpy

Es un módulo de Python encargado de agregar toda la capacidad matemática, vectorial y matricial para realizar operaciones con datos, vectores y matrices. Con este módulo se puede acceder a las funciones matemáticas a alto nivel. En el sistema desarrollado se ha utilizado para el manejo de arrays y listas.

7.4.4 Pandas

Python Data Analysis Library es una biblioteca de Python para la manejar y analizar datos. Proporciona estructuras de datos de alto rendimiento y fáciles de usar, además de diversas herramientas para realizar el análisis de datos. En la implantación se ha utilizado para el manejo de tablas numéricas.

7.4.5 Scikit-learn

Biblioteca de Python para el manejo de Machine Learning, integra gran diversidad de algoritmos para realizar minería de datos. Se ha utilizado para obtener gran parte de las metricas de los modelos, así como los datos de algunas gráficas realizadas.

7.4.6 Matplotlib

Biblioteca de Python para la generación de gráficos a partir de datos contenidos en estructuras Numpy. Todas las gráficas e imágenes obtenidas en los experimentos se han realizado mediante la utilización de esta biblioteca.

7.5 Estructura y código de la implementación

Para obtener una descripción de la estructura del sistema y algunos detalles de implementación, así como detalles sobre cómo poner en funcionamiento el sistema puede leer el **¡Error! No se encuentra el origen de la referencia.**.

8. Experimentos realizados

En este apartado se llevará a cabo la descripción de los experimentos que se realizará con el objetivo de encontrar la arquitectura de CNN que proporcione los mejores resultados para la tarea de reconocimiento de caracteres en imágenes de escenas reales o ambientes duros.

Se hará uso del estudio teórico realizado previamente, tratándose en gran medida de realizar pruebas sobre algunas las arquitecturas de CNN vistas, llevando a cabo diversas modificaciones sobre ellas con el objetivo de mejorarlas de acuerdo con las buenas prácticas definidas en la literatura actual.

Los parámetros que van a variar en los experimentos son:

Arquitectura o modelo de CNN

Se trata de la estructura de capas que se conforma para configurar la red. Las arquitecturas pueden variar en función de diversos parámetros que dependen del tipo de capa que se agregue a la estructura. Los parámetros que pueden variar para los distintos tipos de capas y con los que vamos a experimentar son:

- **Número de capas**: es uno de los parámetros de experimentación que más repercuten en el experimento, depende del conjunto de imágenes utilizado, imágenes más complejas necesitarán más capas e imágenes más sencillas menos.
- **Capa de convolución**: pueden variar el **número de filtros** utilizados y el **tamaño del kernel** utilizado para la aplicación del filtro.
- **Capa de pooling** o submuestreo: puede variar el algoritmo utilizado para reducir las dimensiones de la capa, en este plan de experimentación se va a experimentar con el **Max** y el **Average**. Y, además, el **tamaño** del **pool**.
- **Capa de activación:** puede variar dependiendo de la función de activación elegida, pueden ser de tipo **Sigmoide, Tangente hiperbólica** o **ReLu.**
- **Capa de salida:** al igual que la capa de activación varía en función de la función que se utilice. Se suelen utilizar la función **Tangente hiperbólica** o **Softmax**.

Hiperparámetros

- **Optimizador:** Se trata de la función de optimización utilizada para obtener el mejor rendimiento de la función objetivo. En los inicios se utilizaba el algoritmo **SGD**, pero en la actualidad se ha dejado de usar tanto debido a la aparición del algoritmo **Adam**, que ha mejorado el rendimiento de los modelos de CNNs. No obstante, se experimentará con ambos para poder observar si dicha mejora teórica se aplica en nuestro problema.
- **Learning rate:** es la tasa de aprendizaje que controla cuánto estamos ajustando los pesos de nuestra red con respecto al gradiente de pérdida. Cuanto más bajo sea el valor, más lento converge y se asegura no perder ningún mínimo local, aunque también puede significar el atasco en un mínimo local.

- **Epochs:** se trata de una etapa en la que la red realiza la fase de entrenamiento y validación con todo el conjunto de datos

Conjuntos de datos

Se van a utilizar los dos conjuntos de datos presentados anteriormente. Además de la obviedad que se observa respecto a que un conjunto es más amplio en cuanto al número de distintas clases a clasificar, también se pretende evaluar si el preprocesamiento realizado en el **CJ2** mejora o empeora el rendimiento respecto de **CJ1**.

De este modo y como ya se comentó en las conclusiones del estudio teórico, se comenzará con una prueba inicial de la arquitectura denominada LeNet-5, la cual fue pionera para la tarea de reconocimiento de caracteres y en la que se han basado la gran mayoría de los estudios posteriores, continuando la fase de experimentación en base al estudio teórico y a los resultados obtenidos para la clasificación de caracteres en los conjuntos de datos CJ1 y CJ2.

8.1 Fase inicial: Experimentos base

En esta primera fase, se realizarán experimentos sobre las redes vistas en el estudio teórico con el objetivo de elegir aquellas alternativas que mejores resultados obtengan y sobre las cuales se realizarán modificaciones en aras de mejorar su rendimiento y obtener una solución final.

8.1.1 Experimentos con la red LeNet-5 Original

Esta arquitectura, presentada en el estudio [Lecun1998], significo una nueva línea de investigación para la tarea de reconocimiento de dígitos, dando grandes resultados gracias a las características que presentan las capas convolucionales y de submuestreo. Es por eso, que realizaremos una experimentación probando inicialmente la red original y posteriormente realizando algunas modificaciones para cada uno de los conjuntos de datos, pudiendo posteriormente comparar estos entre sí.

Arquitectura: [Lecun1998]

- Numero de capas: **7 capas**
- Capas de convolución
  - Número de filtros: **6, 16,120**
  - Tamaño kernel: **5**
- Capas de pooling
  - Función: **Average**
  - Tamaño: **2**
- Capas de activación
  - Función: **Sigmoide**
- Capa de salida:
  - Función: **Tangente hiperbólica**

Hiperparámetros

- Función de optimización: **SGD**
- Learning rate: **1.0x10^-3**
- Epochs: **200**

Resultados

Para evaluar los resultados del rendimiento de un experimento se va a tener de referencia las métricas de evaluación descritas en el apartado de este documento denominado Métricas de evaluación, para los dos conjuntos de datos presentados se han obtenido los siguientes resultados.

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje (entrenamiento y validación)**.
  Durante la fase de aprendizaje la red propuesta obtuvo para la métrica de Accuracy la evolución mostrada en la Ilustración 39 y para la función de pérdida o Loss la evolución mostrada en la Ilustración 40.

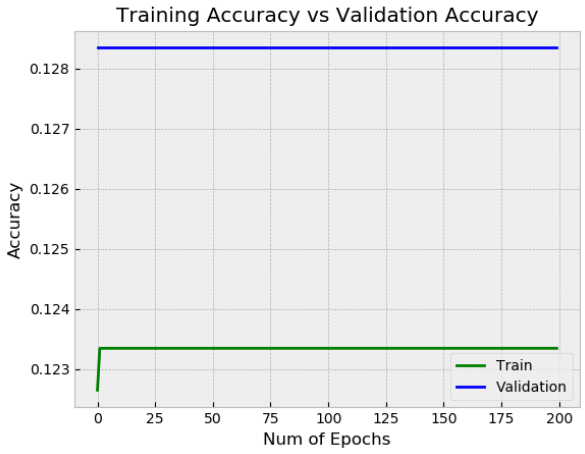

Ilustración 39: Evolución del Accuracy en la fase de aprendizaje con LeNet-5 y CJ1

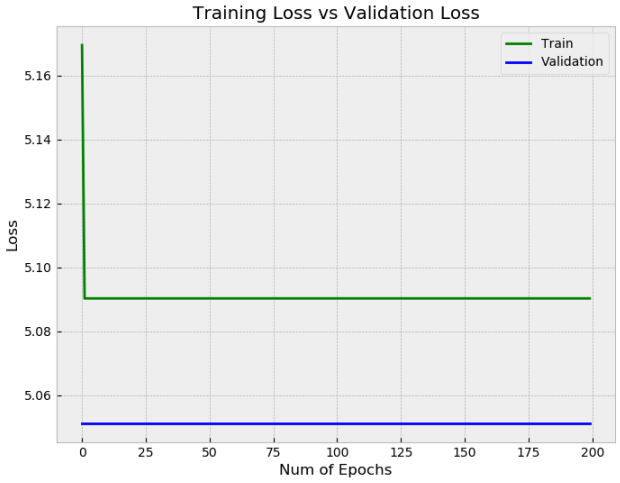

Ilustración 40: Evolución del Loss en la fase de aprendizaje con LeNet-5 y CJ1

Como se puede observar, ya desde la fase de aprendizaje, este modelo no presenta un buen rendimiento para el conjunto de datos, ya que la red no converge y se mantiene en unos valores muy bajos para las métricas de Accuracy y muy altos para la función de pérdida.

- **Fase de testing**.
  Una vez que el modelo ya ha obtenido los pesos de la red resultado de la fase de aprendizaje, se realiza un testing con imágenes desconocidas por la red, donde se evaluará el rendimiento de ésta. Los resultados obtenidos en forma de matriz de confusión para la clasificación de dicho conjunto con la red propuesta se muestra en la Ilustración 41.

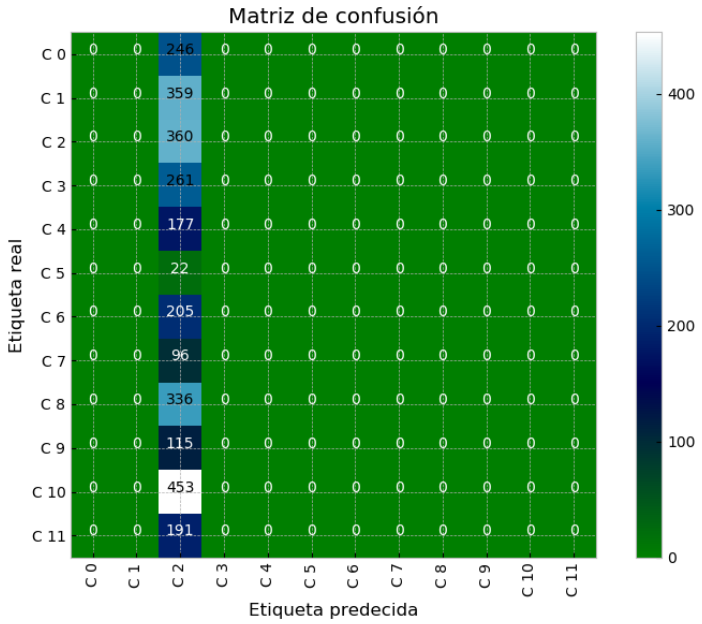

Ilustración 41: Matriz de confusión de la red LeNet-5 con el conjunto de datos CJ1

Los resultados que arroja la matriz, es que la red propuesta no realiza un aprendizaje correcto, ya que clasifica todas las imágenes como pertenecientes a la clase 2 (C2).
Por otra parte, basándonos en los resultados de la matriz se pueden conseguir las métricas de evaluación de la etapa de testing, verificando de este modo los resultados obtenidos en la fase de aprendizaje.
Los resultados obtenidos de este experimento para cada clase de manera individual son los mostrados en la Ilustración 42, y los resultados globales del conjunto, obtenidos mediante la aplicación de la media aritmética de todas las clases para las métricas más características son los mostrados en la Ilustración 43.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	0.0	2575.0	0.0	246.0	0.9128	0.0	1.0	nan	0.9128	0.0	1.0	nan	nan
C 1	0.0	2462.0	0.0	359.0	0.8727	0.0	1.0	nan	0.8727	0.0	1.0	nan	nan
C 2	360.0	0.0	2461.0	0.0	0.1276	1.0	0.0	0.1276	nan	1.0	0.0	0.8724	0.2263
C 3	0.0	2560.0	0.0	261.0	0.9075	0.0	1.0	nan	0.9075	0.0	1.0	nan	nan
C 4	0.0	2644.0	0.0	177.0	0.9373	0.0	1.0	nan	0.9373	0.0	1.0	nan	nan
C 5	0.0	2799.0	0.0	22.0	0.9922	0.0	1.0	nan	0.9922	0.0	1.0	nan	nan
C 6	0.0	2616.0	0.0	205.0	0.9273	0.0	1.0	nan	0.9273	0.0	1.0	nan	nan
C 7	0.0	2725.0	0.0	96.0	0.966	0.0	1.0	nan	0.966	0.0	1.0	nan	nan
C 8	0.0	2485.0	0.0	336.0	0.8809	0.0	1.0	nan	0.8809	0.0	1.0	nan	nan
C 9	0.0	2706.0	0.0	115.0	0.9592	0.0	1.0	nan	0.9592	0.0	1.0	nan	nan
C 10	0.0	2368.0	0.0	453.0	0.8394	0.0	1.0	nan	0.8394	0.0	1.0	nan	nan
C 11	0.0	2630.0	0.0	191.0	0.9323	0.0	1.0	nan	0.9323	0.0	1.0	nan	nan

Ilustración 42: Métricas de evaluación de la etapa de testing del modelo LeNet-5 para CJ1

	TOTAL
ACC	0.8546
PPV	nan
TPR	0.0833
F1	nan

Ilustración 43: Valores globales de las métricas más características del modelo LeNet-5 para CJ1

Realizando una interpretación de los resultados obtenidos, se pueden observar que dependiendo de que métrica se utilice los valores no reflejan de manera correcta la tendencia del modelo. Es decir, en la Ilustración 42, en la cual aparecen las métricas para cada clase, si se tiene como referencia la métrica de Accuracy, se puede ver que en la mayoría de las clases, el valor parece ser bueno, estando en un rango de 83%-95%, significando esto que el modelo tiene una exactitud notable, no obstante, en realidad esto no ocurre.

El motivo por el cual los resultados del Accuracy de las clases no reflejan la situación real del modelo se debe al número de TN y FP para cada clase. Y es que, al ser el valor de TN alto para cada clase, ya que solo se tiene en cuenta las instancias de dicha clase en las que la red se equivoca al clasificar como otra distinta y al ser el valor de FP bajo, entonces, los errores de las demás clases no influyen al ser un problema multiclase. Solo tiene en cuenta como mal clasificados para una clase determinada las instancias de FN.

Si se analiza un caso concreto, se puede entender de manera sencilla. Por ejemplo, la clase 0 de la Ilustración 42. Esta clase tiene los siguientes indicadores de clasificación:

- TP = 0, es decir no clasifica ninguna instancia de la clase 0 correctamente.
- TN = 2575, los valores que no son de la clase 0 los clasifica correctamente, para esta clase concreta.
- FP = 0, no clasifica como clase 0 a instancias que no lo son.
- FN = 246, las instancias de la clase 0 mal clasificadas.

Por lo tanto, el valor del Accuracy para la clase 0 (cuya formula es la vista en (5)), es finalmente un 91.28%, ya que no clasifica a otras clases que no lo son como clase 0, pero la realidad es que tampoco clasifica ninguna instancia de la clase 0 correctamente y esto no es reflejado en dicha métrica.

Es por eso, que en este modelo en el que se clasifican todas las instancias como una clase determinada, en este caso la clase 2, solo hay que analizar dicha clase, ya que ella, si refleja el rendimiento de la red. Si se realiza el análisis de las métricas obtenidas para la clase 2, se puede ver que el valor del Accuracy es de 12,76% y el de precisión de 12,76% también, siendo valores representativos del mal funcionamiento de la red.

Con este análisis, lo que se quiere mostrar es que las métricas obtenidas en base a la matriz de confusión de un modelo concreto, a simple vista pueden no representar la realidad, sino que hay que interpretar los resultados adecuadamente. Ya que para determinados modelos hay que centrar la atención en las métricas de una clase determinada o en una métrica determinada.

En el caso de este experimento, hay que centrarse en la clase 2, ya que otras métricas como TPR (recall) y TNR (specificity) tampoco reflejan el mal rendimiento del modelo. Motivo por el cual obtener una métrica global partiendo de dichas métricas no es correcto, como se puede ver en la Ilustración 43.

Por eso, también se obtuvieron los valores para dichas métricas teniendo en cuenta el balanceado del conjunto de datos. Estos resultados se pueden observar en la Ilustración 44 y en la Ilustración 45, donde sí se reflejan el rendimiento del modelo para las métricas de Precisión, Recall, F1 y Accuracy. Estas métricas no dan una información precisa de los valores de cada clase, sino una visión general.

	Precision	Recall	F1
C 0	0.0	0.0	0.0
C 1	0.0	0.0	0.0
C 2	0.1276	1.0	0.2263
C 3	0.0	0.0	0.0
C 4	0.0	0.0	0.0
C 5	0.0	0.0	0.0
C 6	0.0	0.0	0.0
C 7	0.0	0.0	0.0
C 8	0.0	0.0	0.0
C 9	0.0	0.0	0.0
C 10	0.0	0.0	0.0
C 11	0.0	0.0	0.0

Ilustración 44: Métricas que reflejan el rendimiento del modelo LeNet-5 para CJ1

ACC	0.1276

Ilustración 45: Acurracy representativo del modelo LeNet-5 para CJ1

Otras métricas indicativas del rendimiento de la red son la curva ROC y el AUC, mediante estos se puede ver el umbral del modelo para la exactitud global, así como la capacidad del modelo de predecir una mayor puntuación para instancias positivas en comparación con instancias negativas. En este caso, los resultados que se obtienen para la curva ROC y el valor del AUC sí que refleja el rendimiento real de modelo. Los resultados obtenidos para cada una de las clases de este experimento son los mostrados en la Ilustración 46 para **C0**, la Ilustración 47 para **C1**, la Ilustración 48 para

C2, la Ilustración 49 para **C3**, la Ilustración 50 para **C4**, la Ilustración 51 para **C5**, la Ilustración 52 para **C6**, la Ilustración 53 para **C7**, la Ilustración 54 para **C8**, la Ilustración 55 para **C9**, la Ilustración 56 para **C10** y la Ilustración 57 para **C11**.

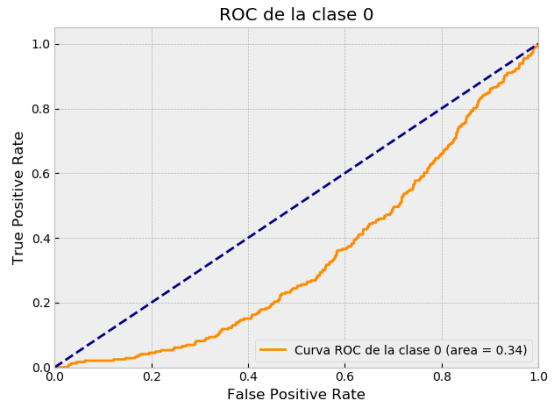

Ilustración 46: Curva ROC y AUC de C0

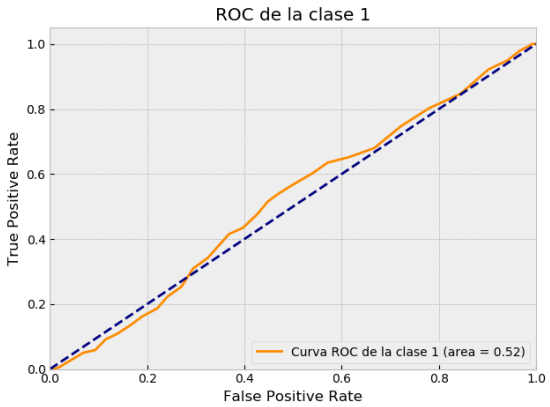

Ilustración 47: Curva ROC y AUC de C1

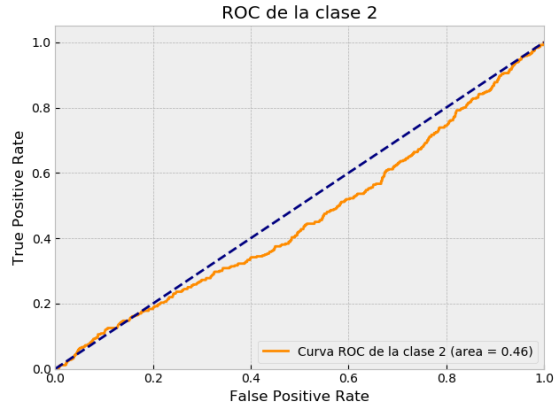

Ilustración 48: Curva ROC y AUC de C2

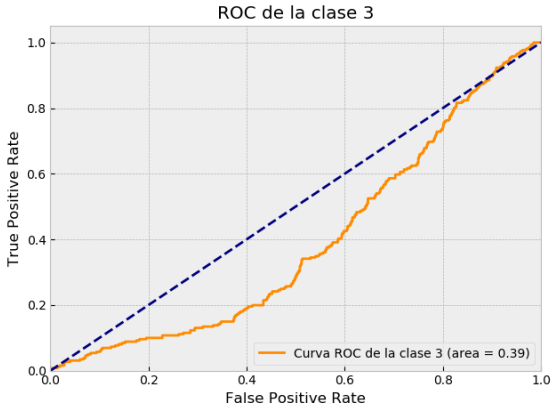

Ilustración 49: Curva ROC y AUC de C3

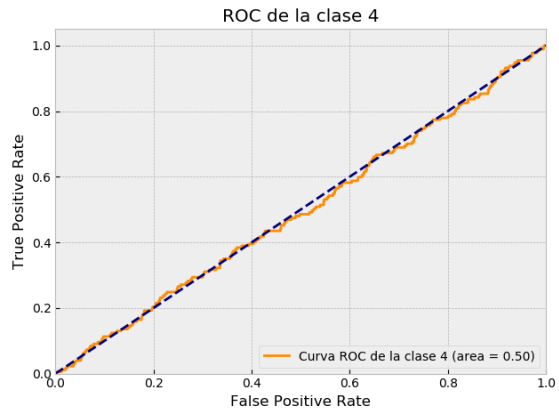

Ilustración 50: Curva ROC y AUC de C4

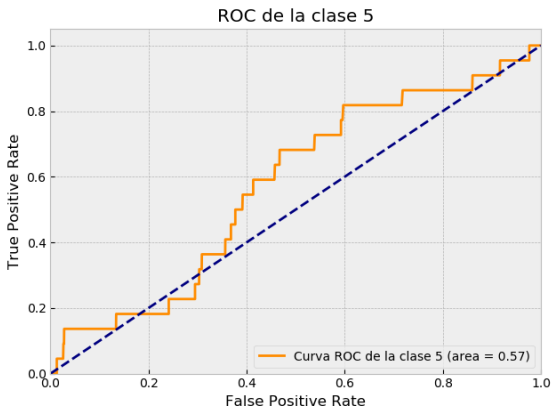

Ilustración 51: Curva ROC y AUC de C5

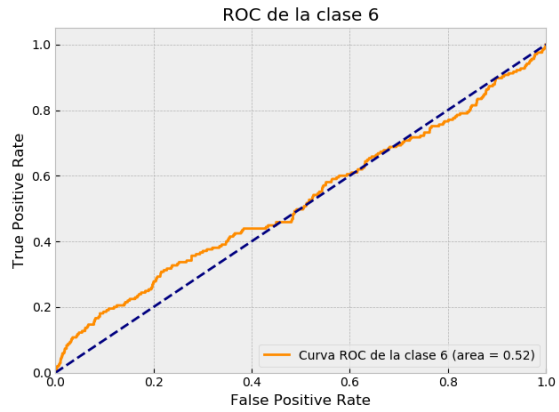

Ilustración 52: Curva ROC y AUC de C6

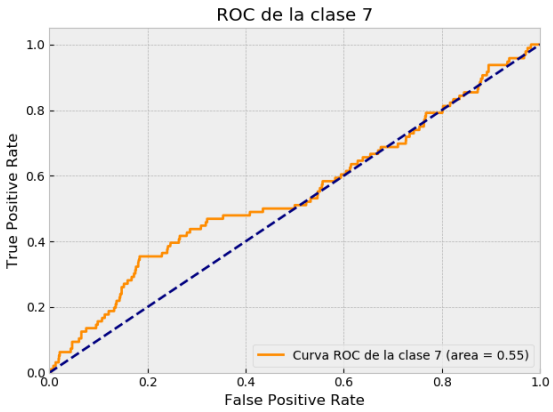

Ilustración 53: Curva ROC y AUC de C7

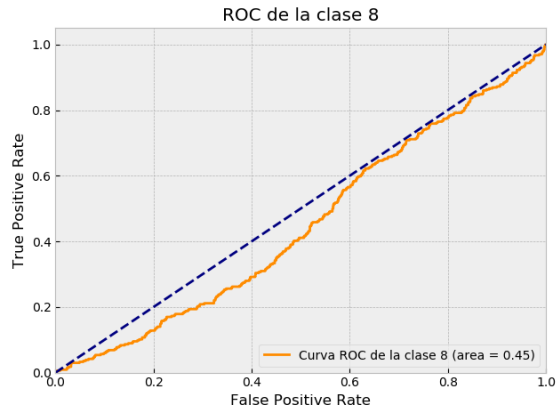

Ilustración 54: Curva ROC y AUC de C8

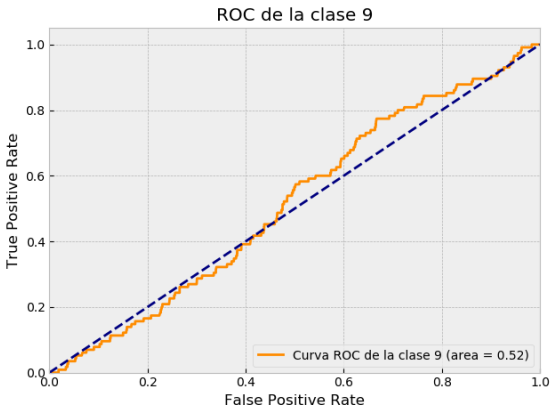

Ilustración 55: Curva ROC y AUC de C9

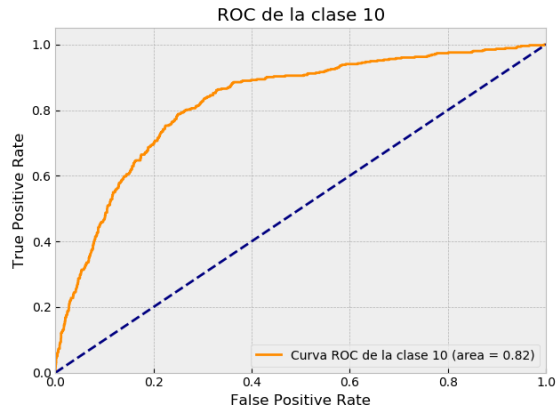

Ilustración 56: Curva ROC y AUC de C10

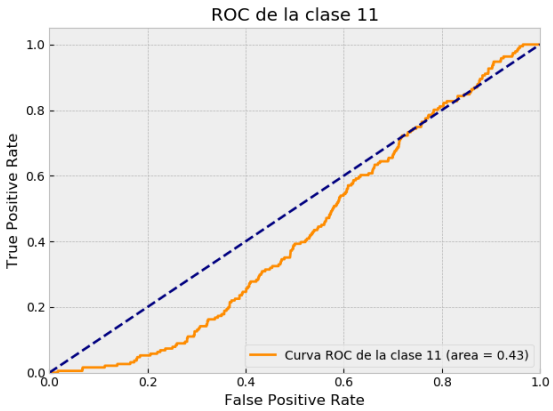

Ilustración 57: Curva ROC y AUC de C11

Tanto gráficamente como por el valor numérico del AUC obtenido para cada clase, se observa que el modelo no es bueno. Situación que se mantiene si calculamos un valor global del modelo como se muestra en la Ilustración 58.

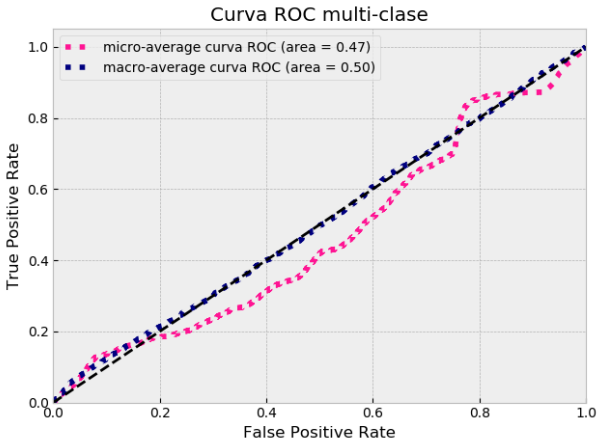

Ilustración 58: Curva ROC y AUC del modelo LeNet-5 para CJ1

- **Conclusiones**

  Los resultados que obtiene el modelo LeNet-5 no son buenos, ya que la red no está aprendiendo las características relevantes de las imágenes y por tanto en ningún momento se produce aprendizaje ni muestras de él, aunque se realicen más epochs.
  Si se analiza el ratio de error obtenido por la red, se puede afirmar que es realmente malo con un valor de **87.23%**, además de los valores obtenidos para las demás métricas ya presentadas. De este modo, esto puede significar que el conjunto de hiperparámetros y funciones de activación y función de salida no son los adecuados para las imágenes, por eso se realizarán cambios en estos aspectos para los experimentos siguientes.

II. Conjunto de imágenes CJ2

Al igual que con CJ1, se realizará un experimento para el conjunto CJ2 y la red LeNet-5, para poder ver si en este caso las imágenes son uno de los elementos que repercute en el mal rendimiento del conjunto de hiperparámetros y arquitectura de red propuesta.

- **Fase de aprendizaje**

  Los resultados de la evolución del modelo LeNet-5 para el Accuracy se pueden ver en la Ilustración 59 y para la función de pérdida o Loss son los mostrados en la Ilustración 60. En ellas se puede observar que la tendencia de los resultados es muy similar a los obtenidos con el conjunto CJ1, aunque numéricamente sean entendiblemente peores.

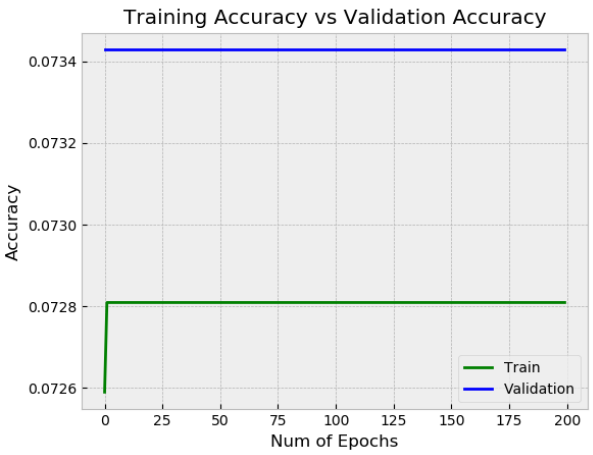

Ilustración 59: Evolución del Accuracy en la fase de aprendizaje con LeNet-5 y CJ2

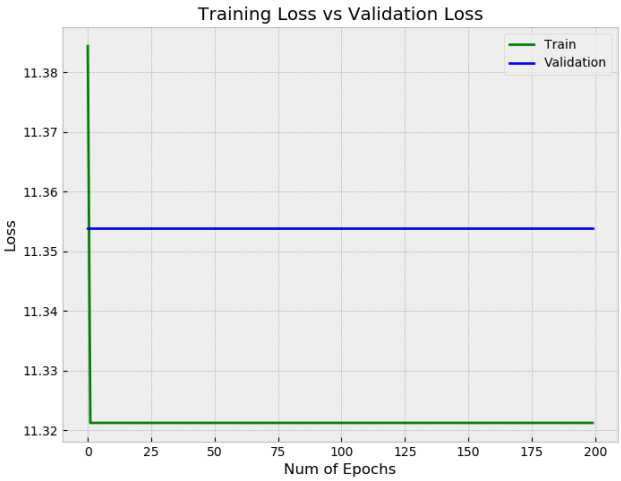

Ilustración 60: Evolución del Loss en la fase de aprendizaje con LeNet-5 y CJ2

- **Fase de testing**

  Como resultado de la fase de evaluación del modelo obtenido se puede observar la Ilustración 61, en ella se muestra la matriz de confusión de dicho modelo. Al igual que con el conjunto de imágenes CJ1, el modelo es pésimo, ya que clasifica todas las imágenes como C2, así, se puede afirmar con certeza que el conjunto de imágenes en este caso no es significativo para la mejora del modelo.

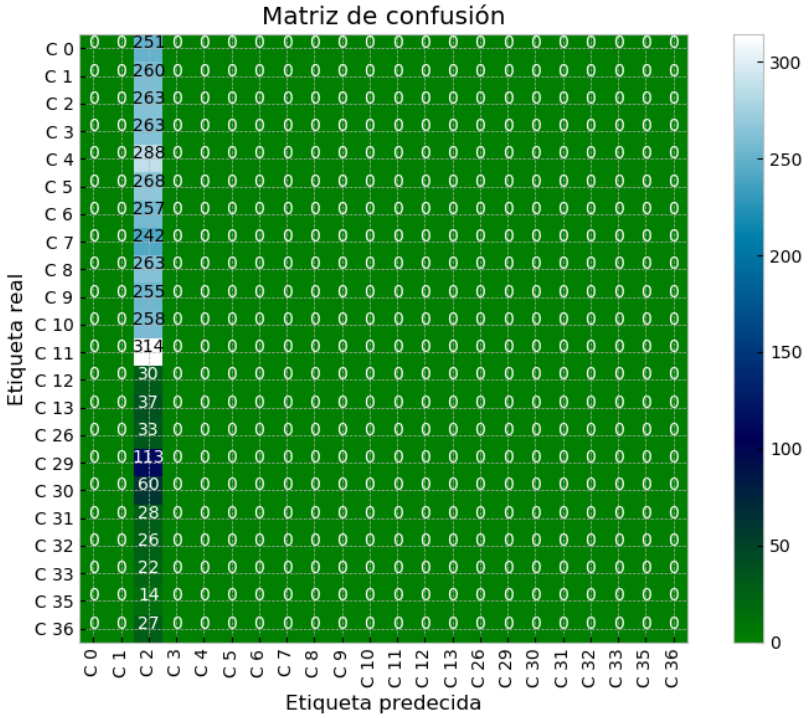

Ilustración 61: Matriz de confusión de la red LeNet-5 con el conjunto de datos CJ2.

Así mismo, las métricas de evaluación obtenidas para dicho experimento también son análogas a CJ1 como se puede observar en la Ilustración 62 y en la Ilustración 63, la única clase cuyos valores son representativos del modelo probado es la clase **C2**.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	0.0	3321.0	0.0	251.0	0.9297	0.0	1.0	nan	0.9297	0.0	1.0	nan	nan
C 1	0.0	3312.0	0.0	260.0	0.9272	0.0	1.0	nan	0.9272	0.0	1.0	nan	nan
C 2	263.0	0.0	3309.0	0.0	0.0736	1.0	0.0	0.0736	nan	1.0	0.0	0.9264	0.1372
C 3	0.0	3309.0	0.0	263.0	0.9264	0.0	1.0	nan	0.9264	0.0	1.0	nan	nan
C 4	0.0	3284.0	0.0	288.0	0.9194	0.0	1.0	nan	0.9194	0.0	1.0	nan	nan
C 5	0.0	3304.0	0.0	268.0	0.925	0.0	1.0	nan	0.925	0.0	1.0	nan	nan
C 6	0.0	3315.0	0.0	257.0	0.9281	0.0	1.0	nan	0.9281	0.0	1.0	nan	nan
C 7	0.0	3330.0	0.0	242.0	0.9323	0.0	1.0	nan	0.9323	0.0	1.0	nan	nan
C 8	0.0	3309.0	0.0	263.0	0.9264	0.0	1.0	nan	0.9264	0.0	1.0	nan	nan
C 9	0.0	3317.0	0.0	255.0	0.9286	0.0	1.0	nan	0.9286	0.0	1.0	nan	nan
C 10	0.0	3314.0	0.0	258.0	0.9278	0.0	1.0	nan	0.9278	0.0	1.0	nan	nan
C 11	0.0	3258.0	0.0	314.0	0.9121	0.0	1.0	nan	0.9121	0.0	1.0	nan	nan
C 12	0.0	3542.0	0.0	30.0	0.9916	0.0	1.0	nan	0.9916	0.0	1.0	nan	nan
C 13	0.0	3535.0	0.0	37.0	0.9896	0.0	1.0	nan	0.9896	0.0	1.0	nan	nan
C 26	0.0	3539.0	0.0	33.0	0.9908	0.0	1.0	nan	0.9908	0.0	1.0	nan	nan
C 29	0.0	3459.0	0.0	113.0	0.9684	0.0	1.0	nan	0.9684	0.0	1.0	nan	nan
C 30	0.0	3512.0	0.0	60.0	0.9832	0.0	1.0	nan	0.9832	0.0	1.0	nan	nan
C 31	0.0	3544.0	0.0	28.0	0.9922	0.0	1.0	nan	0.9922	0.0	1.0	nan	nan
C 32	0.0	3546.0	0.0	26.0	0.9927	0.0	1.0	nan	0.9927	0.0	1.0	nan	nan
C 33	0.0	3550.0	0.0	22.0	0.9938	0.0	1.0	nan	0.9938	0.0	1.0	nan	nan
C 35	0.0	3558.0	0.0	14.0	0.9961	0.0	1.0	nan	0.9961	0.0	1.0	nan	nan
C 36	0.0	3545.0	0.0	27.0	0.9924	0.0	1.0	nan	0.9924	0.0	1.0	nan	nan

Ilustración 62: Métricas que reflejan el rendimiento del modelo LeNet-5 para CJ2

ACC	0.0736

Ilustración 63: Accuracy LeNet-5 para CJ2

Por último, observando la Ilustración 64 donde se ve la gráfica resultante del cálculo de la curva ROC y valor del AUC para el modelo LeNet-5 se puede ver que no es un modelo bueno.

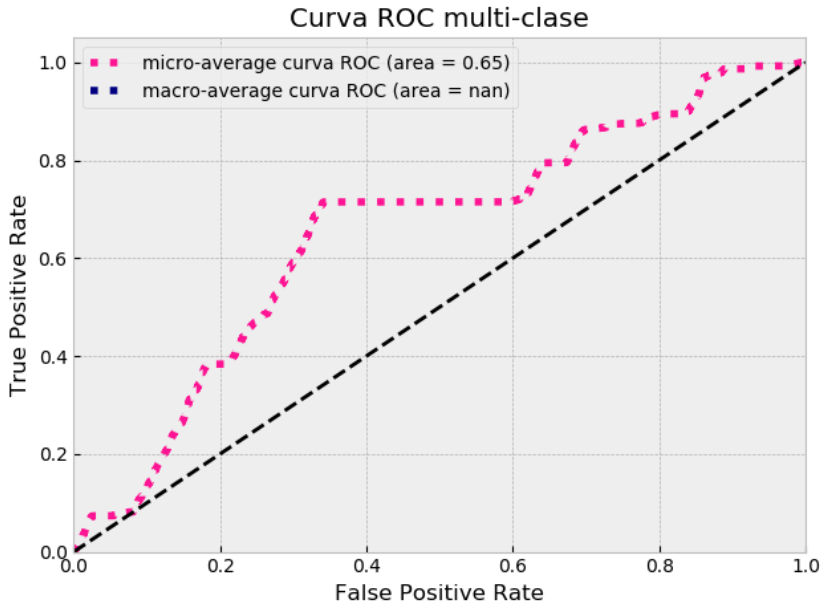

Ilustración 64: Curva ROC y AUC multi-clase de la red LeNet 5 para CJ2

- **Conclusiones**

  En el caso de CJ2, el ratio de error tiene un valor de **92.63%**, de modo que se verifica que el modelo inicial no es adecuado para ninguno de los dos conjuntos de datos, es evidente que habrá que realizar diversos cambios en dicha arquitectura para actualizarla a los avances actuales, por lo tanto, el primer cambio se llevará a cabo en las funciones elegidas para las capas de activación y la capa de salida, introduciendo aquellas que mejores resultados están brindando.

8.1.2 Experimentos LeNet-5 con cambios en capas de activación y capa de salida (LeNet-5 FunAct)

Arquitectura: [Lecun1998]

- Numero de capas: 7 capas
- Capa convolución
  - Número de filtros: 6, 16,120
  - Tamaño kernel: 5
- Capa de pooling
  - Función: Average
  - Tamaño: 2
- Capa de activación
  - Función: **ReLu**
- Capa de salida:
  - Función: **Softmax**

Hiperparámetros

- Función de optimización: SGD
- Learning rate: 1.0x10^-3
- Epochs: 200

Resultados

Los resultados obtenidos tras el cambio de las funciones de activación y de salida para cada uno de los conjuntos de datos se analizarán a continuación.

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  Los resultados obtenidos tras el cambio de las funciones de activación y salida en la fase de aprendizaje arrojan una mejora significativa en el funcionamiento del modelo de manera general. Como se puede ver en la Ilustración 65 y en la Ilustración 66, los valores para las métricas Accuracy y Loss van mejorando de manera exponencial hasta la epoch 25, luego la mejora es más leve hasta mantenerse en valores constantes significativamente mejores a los obtenidos con la red anterior, es decir, en la fase de validación se ve que el Accuracy se puede encontrar sobre 0.9 y el Loss sobre 0.5. Traduciéndose en una gran mejoría del modelo.

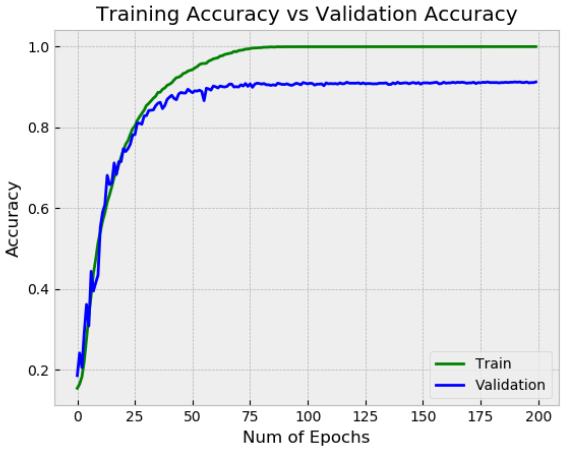

Ilustración 65: Evolución del Accuracy en la fase de aprendizaje con LeNet-5 FunAct y CJ1

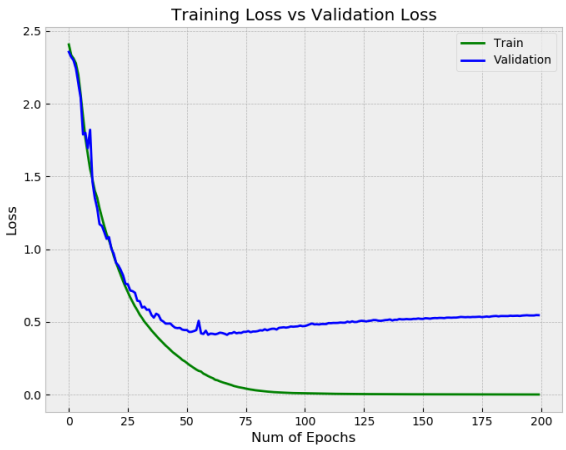

Ilustración 66: Evolución del Loss en la fase de aprendizaje con LeNet-5 FunAct y CJ1

Al verse dicha mejora, se pueden tener en cuenta otras métricas más representativas del rendimiento de la red, como son la Precision (descripción vista en (6)) y el Recall (descripción vista en (7)). De igual modo que ocurre con el Accuracy y el Loss, se ha obtenido la evolución de dichas métricas durante el entrenamiento y la validación, pudiendo observarse en la Ilustración 67 y en la Ilustración 68. Y al igual que con el

Accuracy, los valores obtenidos se encuentran en un intervalo satisfactorio, entrando en los 0.9.

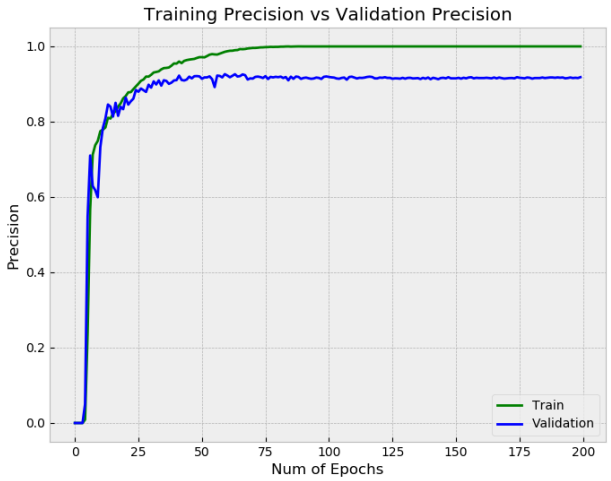

Ilustración 67: Evolución de la Precision en la red LeNet-5 FunAct para CJ1

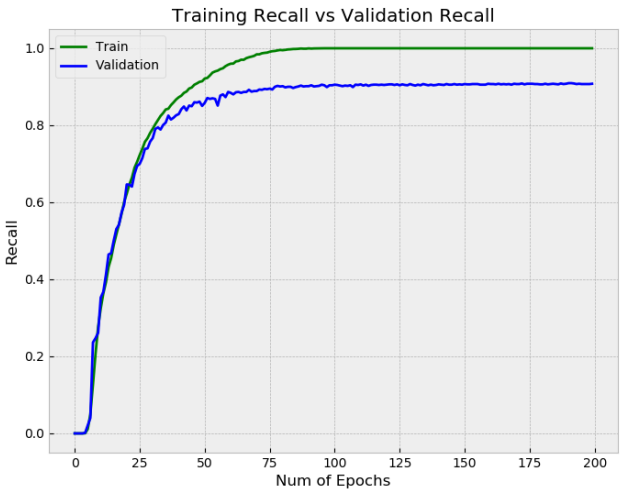

Ilustración 68: Evolución del Recall en la red LeNet-5 FunAct para CJ1

- **Fase de testing**

  Teniendo en cuenta los resultados obtenidos en la fase de aprendizaje, lo normal, sería que la matriz de confusión haya sufrido una gran mejora respecto a la red anterior, teniendo en este caso la diagonal principal mucho más poblada, debido al número de aciertos en la clasificación realizada.

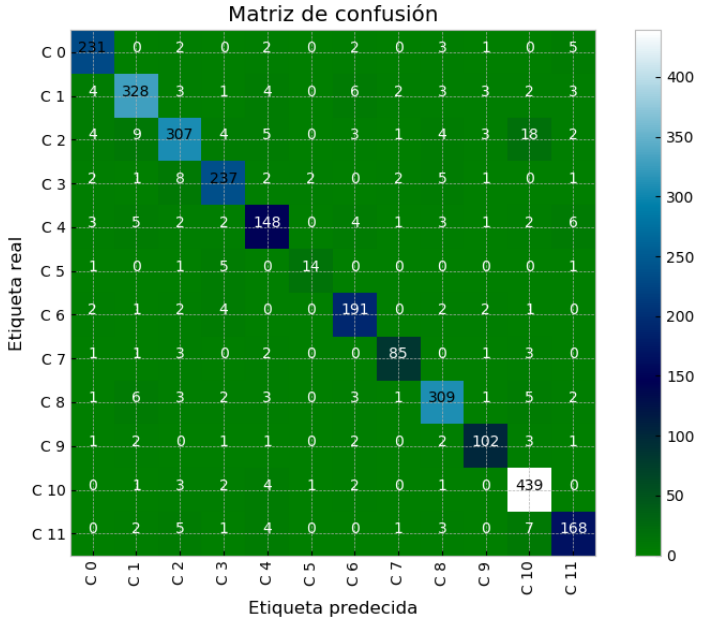

Ilustración 69: Matriz de confusión de la red LeNet-5 FunAct para CJ1

Y en efecto, como se puede observar en la Ilustración 69, la apariencia de la matriz ha mejorado en gran escala debido a que la red clasifica de manera correcta gran parte del conjunto de test.

No obstante, para realizar una interpretación más precisa de los resultados obtenidos por la red, se pueden analizar los valores obtenidos para cada clase como se muestra en la Ilustración 70 y en la Ilustración 71. En este caso, se ve de manera clara que la red no realiza una clasificación basada en el azar o eligiendo una sola clase, sino que realmente tiene en cuenta las características más representativas de cada carácter y trabaja en función de ellas.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	231.0	2556.0	19.0	15.0	0.9879	0.939	0.9926	0.924	0.9942	0.0074	0.061	0.076	0.9315
C 1	328.0	2434.0	28.0	31.0	0.9791	0.9136	0.9886	0.9213	0.9874	0.0114	0.0864	0.0787	0.9175
C 2	307.0	2429.0	32.0	53.0	0.9699	0.8528	0.987	0.9056	0.9786	0.013	0.1472	0.0944	0.8784
C 3	237.0	2538.0	22.0	24.0	0.9837	0.908	0.9914	0.9151	0.9906	0.0086	0.092	0.0849	0.9115
C 4	148.0	2617.0	27.0	29.0	0.9801	0.8362	0.9898	0.8457	0.989	0.0102	0.1638	0.1543	0.8409
C 5	14.0	2796.0	3.0	8.0	0.9961	0.6364	0.9989	0.8235	0.9971	0.0011	0.3636	0.1765	0.7179
C 6	191.0	2594.0	22.0	14.0	0.9872	0.9317	0.9916	0.8967	0.9946	0.0084	0.0683	0.1033	0.9139
C 7	85.0	2717.0	8.0	11.0	0.9933	0.8854	0.9971	0.914	0.996	0.0029	0.1146	0.086	0.8995
C 8	309.0	2459.0	26.0	27.0	0.9812	0.9196	0.9895	0.9224	0.9891	0.0105	0.0804	0.0776	0.921
C 9	102.0	2693.0	13.0	13.0	0.9908	0.887	0.9952	0.887	0.9952	0.0048	0.113	0.113	0.887
C 10	439.0	2327.0	41.0	14.0	0.9805	0.9691	0.9827	0.9146	0.994	0.0173	0.0309	0.0854	0.9411
C 11	168.0	2609.0	21.0	23.0	0.9844	0.8796	0.992	0.8889	0.9913	0.008	0.1204	0.1111	0.8842

Ilustración 70: Métricas que reflejan el rendimiento del modelo LeNet-5 FunAct para CJ1

ACC	0.9071

Ilustración 71: Accuracy de LeNet-5 FunAct para CJ1

Por último, queda por analizar el valor de la curva ROC y el valor del AUC, en este caso, como se puede ver en la Ilustración 72, se obtienen valores muy buenos.

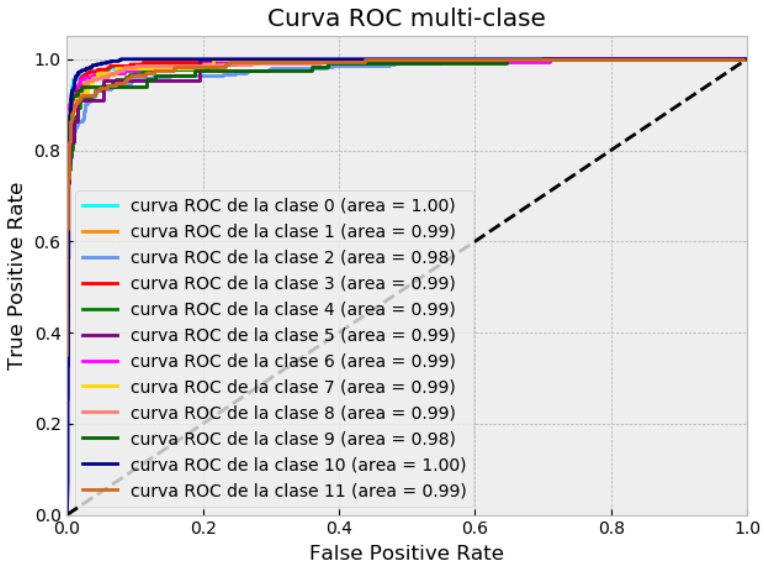

Ilustración 72: Curva ROC y AUC red LeNet-5 FunAct para CJ1

- **Conclusiones**

  Analizando los resultados tras el cambio de las funciones se puede observar que se ha obtenido una gran mejora en el rendimiento de la red, obteniendo un ratio de error de **9.28%**. Esto era de esperarse debido a que la nueva función de activación utilizada ReLu se trata de una función muy sencilla (máximo) y además solo presenta un punto problemático en $x = 0$, mientras que la anterior tenía problema en los valores extremos donde la derivada fuese 0. En cuanto a la función de la capa de salida, la mejora también es significativa, ya que la función Softmax representa una distribución de probabilidad, filtrando de este modo un conjunto de valores que se encuentren debajo de un valor máximo establecido. Además, actualmente no se utiliza otra función como capa final de los clasificadores, debido a su gran desempeño.

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  En el caso de CJ2, la evolución del Accuracy en la fase de aprendizaje se muestra en la Ilustración 73 y la del Loss en la Ilustración 74, y al igual que con CJ1, con dicha red si se observa un proceso de aprendizaje de características tras el paso de las epochs, obteniendo valores sobre 0.9 para el Accuracy y menores que 0.5 para la función de pérdida.

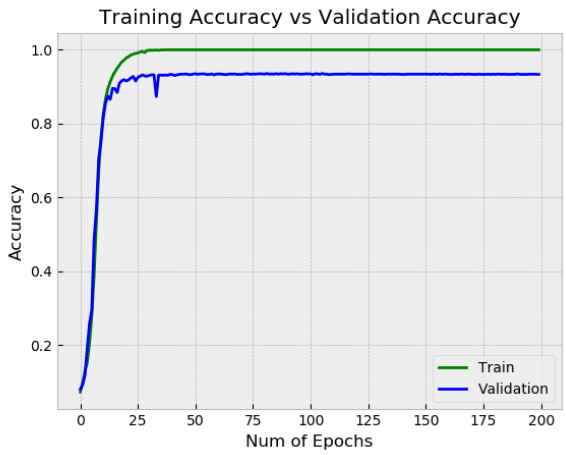

Ilustración 73: Evolución Accuracy en la fase aprendizaje modelo LeNet-5 FunAct para CJ2

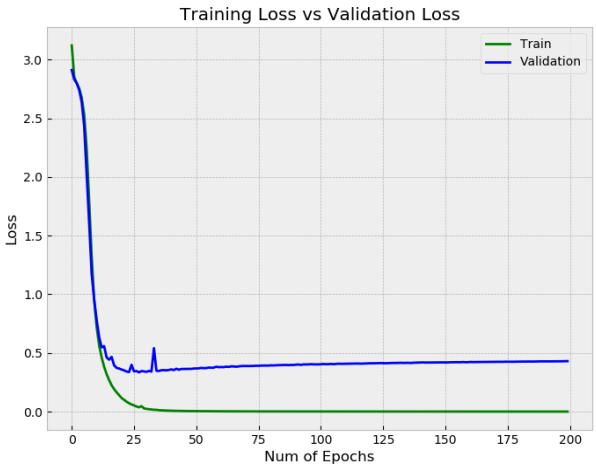

Ilustración 74: Evolución Loss en la fase aprendizaje modelo LeNet-5 FunAct para CJ2

- **Fase de testing**

  Los resultados obtenidos para la matriz de confusión son los mostrados en la Ilustración 75, en ella también es visible la gran mejoría del modelo.

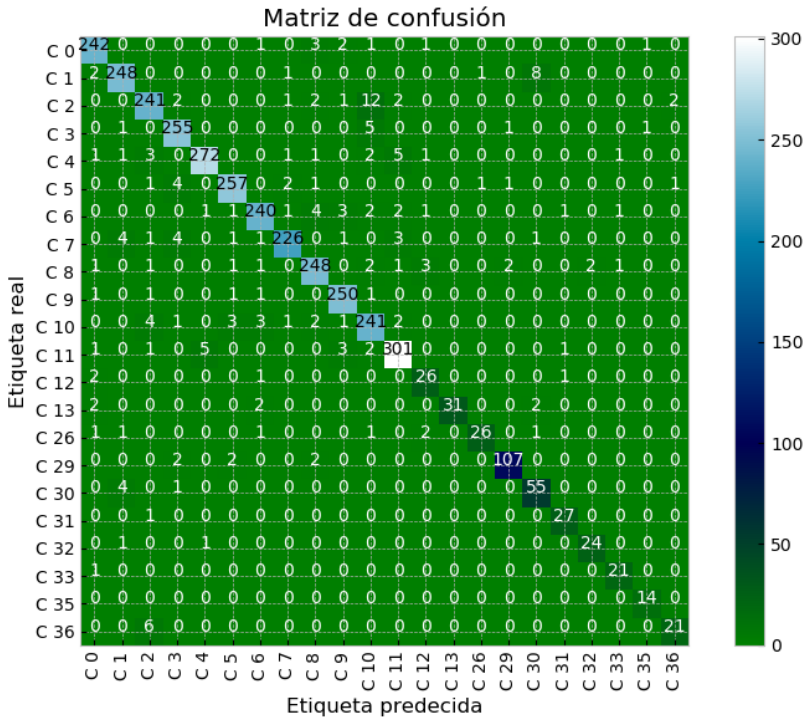

Ilustración 75: Matriz de confusión de la red LeNet-5 FunAct para CJ2

De este modo, los valores obtenidos de las métricas son los mostrados en la Ilustración 76 y el Accuracy final es el mostrado en Ilustración 77.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	242.0	3309.0	12.0	9.0	0.9941	0.9641	0.9964	0.9528	0.9973	0.0036	0.0359	0.0472	0.9584
C 1	248.0	3300.0	12.0	12.0	0.9933	0.9538	0.9964	0.9538	0.9964	0.0036	0.0462	0.0462	0.9538
C 2	241.0	3290.0	19.0	22.0	0.9885	0.9163	0.9943	0.9269	0.9934	0.0057	0.0837	0.0731	0.9216
C 3	255.0	3295.0	14.0	8.0	0.9938	0.9696	0.9958	0.948	0.9976	0.0042	0.0304	0.052	0.9586
C 4	272.0	3277.0	7.0	16.0	0.9936	0.9444	0.9979	0.9749	0.9951	0.0021	0.0556	0.0251	0.9594
C 5	257.0	3295.0	9.0	11.0	0.9944	0.959	0.9973	0.9662	0.9967	0.0027	0.041	0.0338	0.9625
C 6	240.0	3304.0	11.0	17.0	0.9922	0.9339	0.9967	0.9562	0.9949	0.0033	0.0661	0.0438	0.9449
C 7	226.0	3323.0	7.0	16.0	0.9936	0.9339	0.9979	0.97	0.9952	0.0021	0.0661	0.03	0.9516
C 8	248.0	3294.0	15.0	15.0	0.9916	0.943	0.9955	0.943	0.9955	0.0045	0.057	0.057	0.943
C 9	250.0	3306.0	11.0	5.0	0.9955	0.9804	0.9967	0.9579	0.9985	0.0033	0.0196	0.0421	0.969
C 10	241.0	3286.0	28.0	17.0	0.9874	0.9341	0.9916	0.8959	0.9949	0.0084	0.0659	0.1041	0.9146
C 11	301.0	3243.0	15.0	13.0	0.9922	0.9586	0.9954	0.9525	0.996	0.0046	0.0414	0.0475	0.9556
C 12	26.0	3534.0	8.0	4.0	0.9966	0.8667	0.9977	0.7647	0.9989	0.0023	0.1333	0.2353	0.8125
C 13	31.0	3535.0	0.0	6.0	0.9983	0.8378	1.0	1.0	0.9983	0.0	0.1622	0.0	0.9118
C 26	26.0	3537.0	2.0	7.0	0.9975	0.7879	0.9994	0.9286	0.998	0.0006	0.2121	0.0714	0.8525
C 29	107.0	3455.0	4.0	6.0	0.9972	0.9469	0.9988	0.964	0.9983	0.0012	0.0531	0.036	0.9554
C 30	55.0	3500.0	12.0	5.0	0.9952	0.9167	0.9966	0.8209	0.9986	0.0034	0.0833	0.1791	0.8661
C 31	27.0	3541.0	3.0	1.0	0.9989	0.9643	0.9992	0.9	0.9997	0.0008	0.0357	0.1	0.931
C 32	24.0	3544.0	2.0	2.0	0.9989	0.9231	0.9994	0.9231	0.9994	0.0006	0.0769	0.0769	0.9231
C 33	21.0	3547.0	3.0	1.0	0.9989	0.9545	0.9992	0.875	0.9997	0.0008	0.0455	0.125	0.913
C 35	14.0	3556.0	2.0	0.0	0.9994	1.0	0.9994	0.875	1.0	0.0006	0.0	0.125	0.9333
C 36	21.0	3542.0	3.0	6.0	0.9975	0.7778	0.9992	0.875	0.9983	0.0008	0.2222	0.125	0.8235

Ilustración 76: Métricas que reflejan el rendimiento del modelo LeNet-5 FunAct para CJ2

ACC	0.9443

Ilustración 77: Accuracy de LeNet-5 FunAct para CJ2

Por último, los valores de la curva ROC y el AUC se pueden ver en Ilustración 78, en este caso solo se tiene en cuenta el valor de la curva ROC para micro-average, debido a la falta de algunas clases de las que no se tienen caracteres.

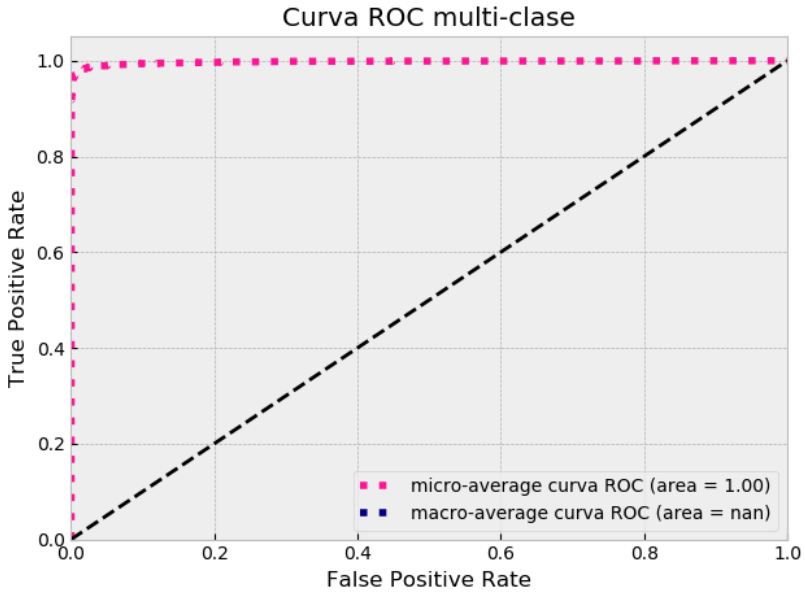

Ilustración 78: Curva ROC y AUC red LeNet-5 FunAct para CJ2

- **Conclusiones**
  Las conclusiones son idénticas a las obtenidas con el experimento realizado para CJ1, la modificación de la función de activación y de salida, han propiciado que el aprendizaje de la red sea fructífero obteniendo un ratio de error de **5.57%**, no obstante, se puede observar que los resultados aún distan en gran medida de los valores obtenidos en el estado de la técnica. Para intentar mejorar esto, se probará cambiando el algoritmo de optimización por el usado en la actualidad.

8.1.3 Experimentos LeNet-5 FunAct con cambio de optimizador (LeNet-5 FunActOpt)

Arquitectura: [Lecun1998]

- Numero de capas: 7 capas
- Capa convolución
  - Número de filtros: 6, 16,120
  - Tamaño kernel: 5
- Capa de pooling
  - Función: Average
  - Tamaño: 2
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: **Adam**
- Learning rate: 1.0x10^-3
- Epochs: 200

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  El resultado obtenido tras el cambio del algoritmo de optimización se puede ver en la evolución de la Ilustración 79 y en la Ilustración 80. A simple vista se puede ver que los valores han mejorado respecto a la red anterior, ya que el valor del Accuracy ahora está mucho más cercano al 1 y el valor del Loss está cerca del 0.2, pero además, también la tendencia de las gráficas han cambiado, en este caso prácticamente en las 10 primeras epoch ya se obtiene unos valores muy buenos, siendo de este modo la etapa de aprendizaje mucho más rápida, siendo innecesario realizar un entrenamiento con 200 epochs, sino muchísimas menos.

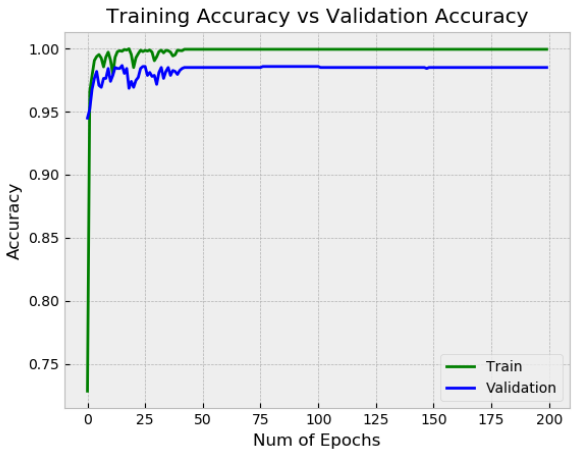

Ilustración 79: Evolución del Accuracy en la fase de aprendizaje con LeNet-5 FunActOpt y CJ1

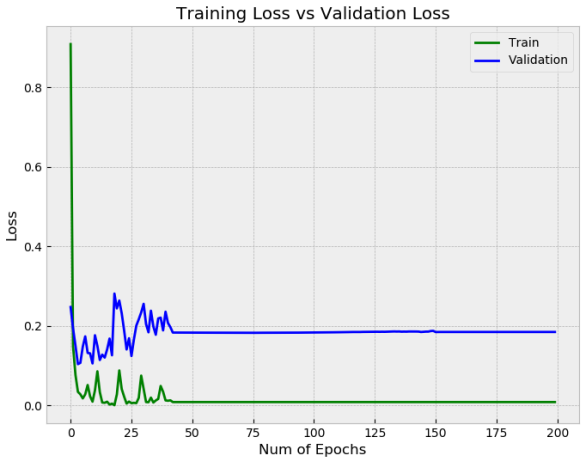

Ilustración 80: Evolución del Loss en la fase de aprendizaje con LeNet-5 FunActOpt y CJ1

- **Fase de testing**

  En cuanto a los resultados del testing, la matriz de confusión es todavía mejor a la presentada en la red anterior, la cantidad de imágenes cuya clasificación ha sido errónea ha sido mucho menor y, por lo tanto, la diagonal principal está más poblada aún como se puede ver en la Ilustración 81.

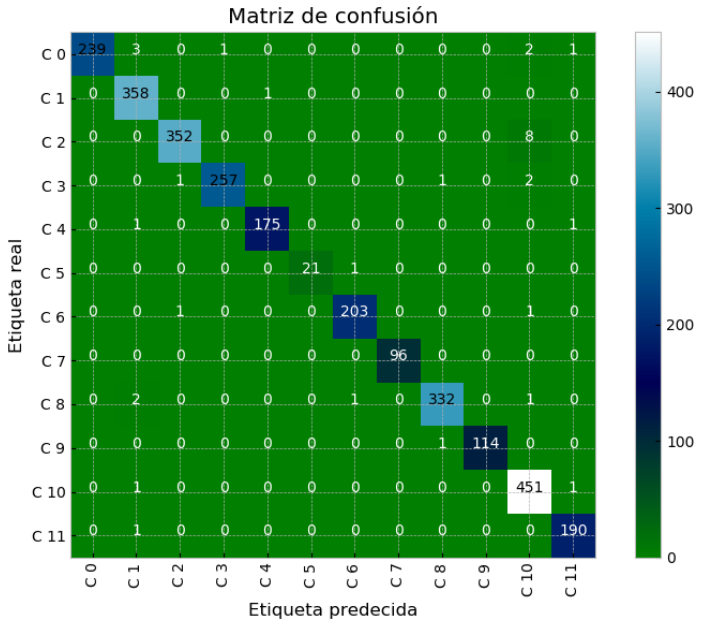

Ilustración 81: Matriz de confusión de la red LeNet-5 FunActOpt para CJ1

Dicha mejoría presentada en la matriz de confusión se ve reflejada en las métricas como muestra la Ilustración 82 y en la Ilustración 83.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	239.0	2575.0	0.0	7.0	0.9975	0.9715	1.0	1.0	0.9973	0.0	0.0285	0.0	0.9856
C 1	358.0	2454.0	8.0	1.0	0.9968	0.9972	0.9968	0.9781	0.9996	0.0032	0.0028	0.0219	0.9876
C 2	352.0	2459.0	2.0	8.0	0.9965	0.9778	0.9992	0.9944	0.9968	0.0008	0.0222	0.0056	0.986
C 3	257.0	2559.0	1.0	4.0	0.9982	0.9847	0.9996	0.9961	0.9984	0.0004	0.0153	0.0039	0.9904
C 4	175.0	2643.0	1.0	2.0	0.9989	0.9887	0.9996	0.9943	0.9992	0.0004	0.0113	0.0057	0.9915
C 5	21.0	2799.0	0.0	1.0	0.9996	0.9545	1.0	1.0	0.9996	0.0	0.0455	0.0	0.9767
C 6	203.0	2614.0	2.0	2.0	0.9986	0.9902	0.9992	0.9902	0.9992	0.0008	0.0098	0.0098	0.9902
C 7	96.0	2725.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 8	332.0	2483.0	2.0	4.0	0.9979	0.9881	0.9992	0.994	0.9984	0.0008	0.0119	0.006	0.991
C 9	114.0	2706.0	0.0	1.0	0.9996	0.9913	1.0	1.0	0.9996	0.0	0.0087	0.0	0.9956
C 10	451.0	2354.0	14.0	2.0	0.9943	0.9956	0.9941	0.9699	0.9992	0.0059	0.0044	0.0301	0.9826
C 11	190.0	2627.0	3.0	1.0	0.9986	0.9948	0.9989	0.9845	0.9996	0.0011	0.0052	0.0155	0.9896

Ilustración 82: Métricas que reflejan el rendimiento del modelo LeNet-5 FunActOpt para CJ1

ACC	0.9883

Ilustración 83: Accuracy de LeNet-5 FunActOpt para CJ1

Por último, en la Ilustración 84 se puede ver el valor de la Curva ROC y del AUC para la red, en este caso el valor es inmejorable.

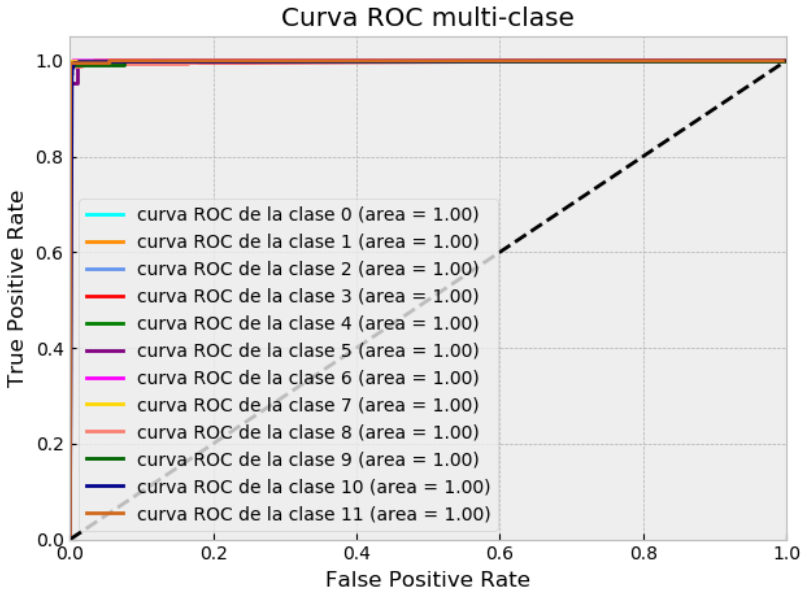

Ilustración 84: Curva ROC y AUC red LeNet-5 FunActOpt para CJ1

- **Conclusiones**

  El cambio realizado en el algoritmo de optimización ha resultado exitoso obteniendo un ratio de error de **1.16%**. Esto se debe a la mejora en el funcionamiento de la red, esto no es una regla general, ya que depende del conjunto de datos y de otros hiperparámetros como el learning rate. Pero la realidad es que en la actualidad se usa de manera más generalizada el algoritmo Adam. Por lo tanto, la mejor forma de saber que algoritmo es mejor para el problema es realizar una prueba, en este caso ha mejorado de manera clara.

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  Para CJ2 la evolución de la red en la fase de aprendizaje ha sido algo más lenta que en el caso de CJ1, pero comparándola con el modelo de red anterior se ha dado una mejoría notable, si se observa la Ilustración 85 y la Ilustración 86, se puede ver que el valor del Accuracy es muy cercano a 1 y el valor de la función de perdida oscila por debajo del 0.3.

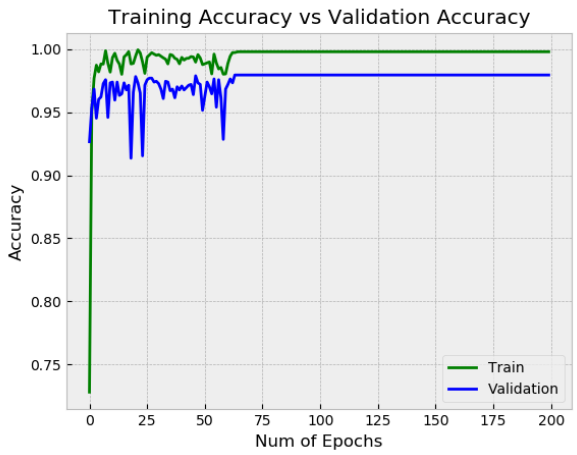

Ilustración 85: Evolución Accuracy en la fase aprendizaje modelo LeNet-5 FunActOpt para CJ2

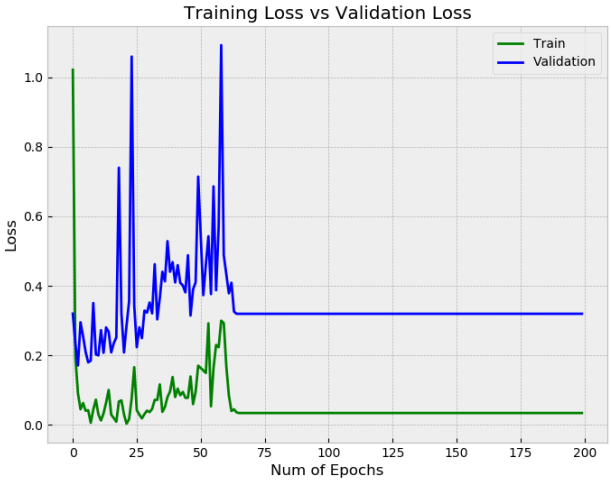

Ilustración 86: Evolución Loss en la fase de aprendizaje modelo LeNet-5 FunActOpt para CJ2

- **Fase de testing**
  Como resultado de la fase de testing se obtiene la matriz de confusión de la Ilustración 87, en ella se visualiza la mejora en la tarea de clasificación ya que las posiciones fuera de la diagonal principal oscilan entre los valores 0,1,2 a excepción de alguna clase.

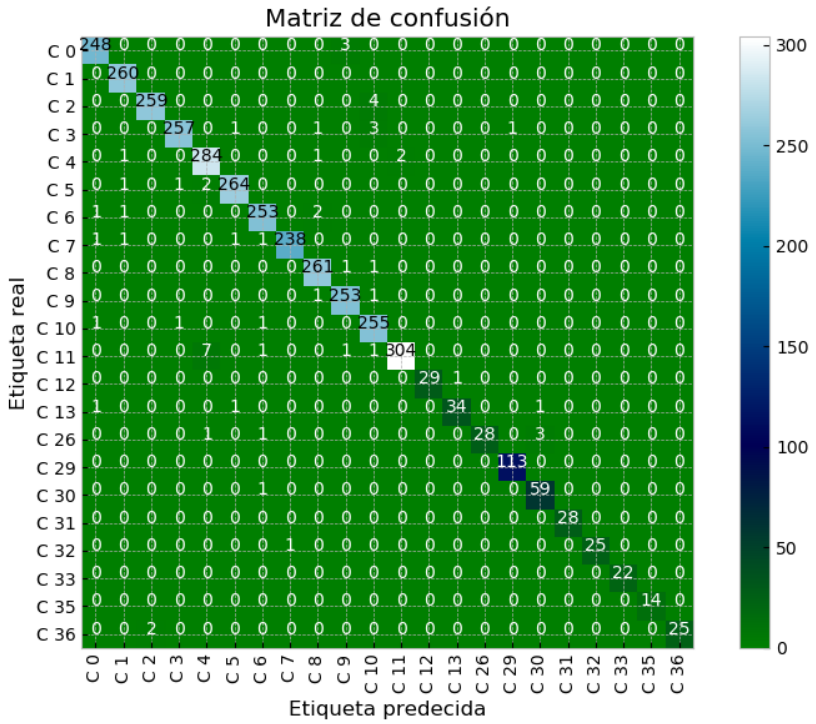

Ilustración 87: Matriz de confusión de la red LeNet-5 FunActOpt para CJ2

Estos resultados traducidos en métricas son los obtenidos en la Ilustración 88 y en la Ilustración 89, como se puede observar han mejorado en gran medida respecto al experimento anterior.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	248.0	3317.0	4.0	3.0	0.998	0.988	0.9988	0.9841	0.9991	0.0012	0.012	0.0159	0.9861
C 1	260.0	3308.0	4.0	0.0	0.9989	1.0	0.9988	0.9848	1.0	0.0012	0.0	0.0152	0.9924
C 2	259.0	3307.0	2.0	4.0	0.9983	0.9848	0.9994	0.9923	0.9988	0.0006	0.0152	0.0077	0.9885
C 3	257.0	3307.0	2.0	6.0	0.9978	0.9772	0.9994	0.9923	0.9982	0.0006	0.0228	0.0077	0.9847
C 4	284.0	3274.0	10.0	4.0	0.9961	0.9861	0.997	0.966	0.9988	0.003	0.0139	0.034	0.9759
C 5	264.0	3301.0	3.0	4.0	0.998	0.9851	0.9991	0.9888	0.9988	0.0009	0.0149	0.0112	0.9869
C 6	253.0	3310.0	5.0	4.0	0.9975	0.9844	0.9985	0.9806	0.9988	0.0015	0.0156	0.0194	0.9825
C 7	238.0	3329.0	1.0	4.0	0.9986	0.9835	0.9997	0.9958	0.9988	0.0003	0.0165	0.0042	0.9896
C 8	261.0	3304.0	5.0	2.0	0.998	0.9924	0.9985	0.9812	0.9994	0.0015	0.0076	0.0188	0.9868
C 9	253.0	3312.0	5.0	2.0	0.998	0.9922	0.9985	0.9806	0.9994	0.0015	0.0078	0.0194	0.9864
C 10	255.0	3304.0	10.0	3.0	0.9964	0.9884	0.997	0.9623	0.9991	0.003	0.0116	0.0377	0.9751
C 11	304.0	3256.0	2.0	10.0	0.9966	0.9682	0.9994	0.9935	0.9969	0.0006	0.0318	0.0065	0.9806
C 12	29.0	3542.0	0.0	1.0	0.9997	0.9667	1.0	1.0	0.9997	0.0	0.0333	0.0	0.9831
C 13	34.0	3534.0	1.0	3.0	0.9989	0.9189	0.9997	0.9714	0.9992	0.0003	0.0811	0.0286	0.9444
C 26	28.0	3539.0	0.0	5.0	0.9986	0.8485	1.0	1.0	0.9986	0.0	0.1515	0.0	0.918
C 29	113.0	3458.0	1.0	0.0	0.9997	1.0	0.9997	0.9912	1.0	0.0003	0.0	0.0088	0.9956
C 30	59.0	3508.0	4.0	1.0	0.9986	0.9833	0.9989	0.9365	0.9997	0.0011	0.0167	0.0635	0.9593
C 31	28.0	3544.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 32	25.0	3546.0	0.0	1.0	0.9997	0.9615	1.0	1.0	0.9997	0.0	0.0385	0.0	0.9804
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	25.0	3545.0	0.0	2.0	0.9994	0.9259	1.0	1.0	0.9994	0.0	0.0741	0.0	0.9615

Ilustración 88: Métricas que reflejan el rendimiento del modelo LeNet-5 FunActOpt para CJ2

ACC	0.9835

Ilustración 89: Accuracy de LeNet-5 FunActOpt para CJ2

Por último, el valor obtenido para la curva ROC y el AUC es el mostrado en la Ilustración 90.

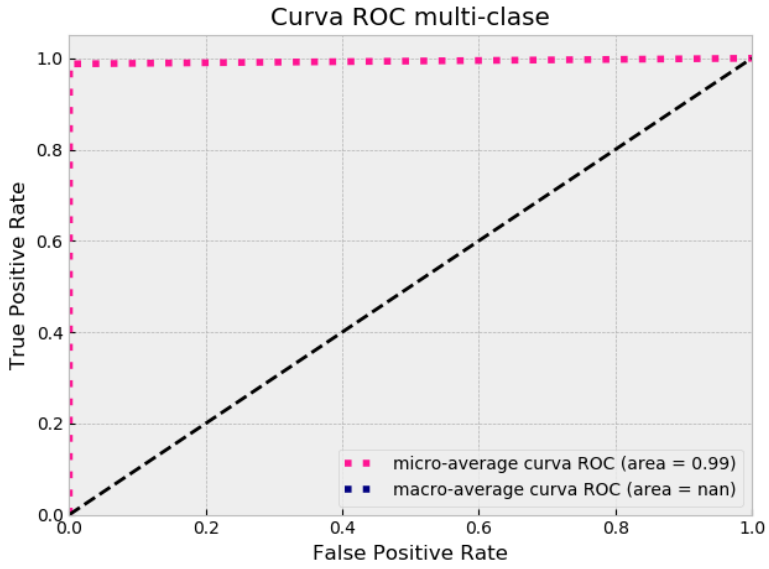

Ilustración 90: Curva ROC y AUC red LeNet-5 FunActOpt para CJ2

- **Conclusiones**
  Las conclusiones obtenidas de este experimento son las mismas obtenidas para CJ1, la elección de Adam como algoritmo de optimización ha mejorado el funcionamiento de la red y agilizado la labor de aprendizaje obteniendo un ratio de error de **1.65%**.

  Finalmente, tras la realización de los experimentos con la red LeNet-5 y los cambios realizados en ella se ha llegado a un resultado donde el valor del error obtenido por la red aún está por encima del 1%, acercándose al 0.95 que se obtenía en el estudio visto, no obstante, hay que tener en cuenta que los dataset utilizados en este caso son de una complejidad más alta.

  Por lo tanto, se seguirá el plan de experimentación con algunas de las restantes redes en las cuales se obtenían mejores resultados, con el objetivo de poder obtener una red base de mejores características y luego poder adaptarla.

8.1.4 Experimentos con la red Ranzato2006

Se trata de la red presentada en el estudio [Ranzato2006] cuya arquitectura es idéntica a LeNet-5 con la salvedad de utilizar un número mayor de filtros en las capas de convolución, obteniendo así más características. Para experimentar con ella, además de modificar el número de filtros también mantendremos los cambios en las funciones de activación y salida, al igual que el optimizador. Otro de los cambios respecto a la batería de experimentos base, es el número de epochs elegido, en los experimentos anteriores se puede ver que para un número inferior de epochs el sistema ya ha realizado toda la fase de aprendizaje, por lo que se reducirá a un valor de 100 epochs.

Arquitectura: [Ranzato2006]

- Numero de capas: 7 capas
- Capa convolución
  - Número de filtros: **50, 50,200**
  - Tamaño kernel: 5
- Capa de pooling
  - Función: Average
  - Tamaño: 2
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: **100**

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

Los resultados obtenidos con la red Ranzato2006 durante la fase de aprendizaje son los mostrados en la Ilustración 91 y en la Ilustración 92. Se puede observar que la red sigue una tendencia similar a la presentada por las últimas redes del experimento anterior, aunque el proceso de aprendizaje parece ser más lento.

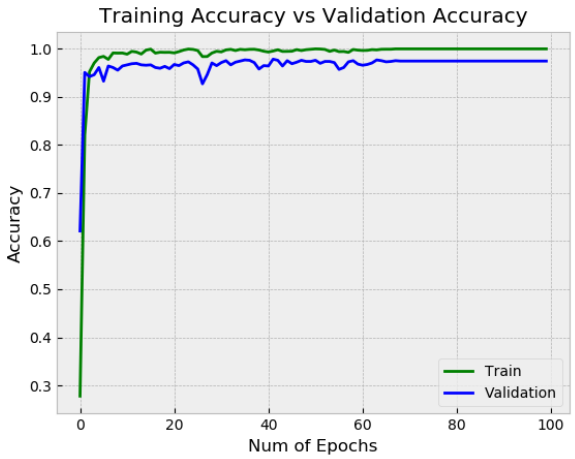

Ilustración 91: Evolución del Accuracy en la fase de aprendizaje con Ranzato2006 y CJ1

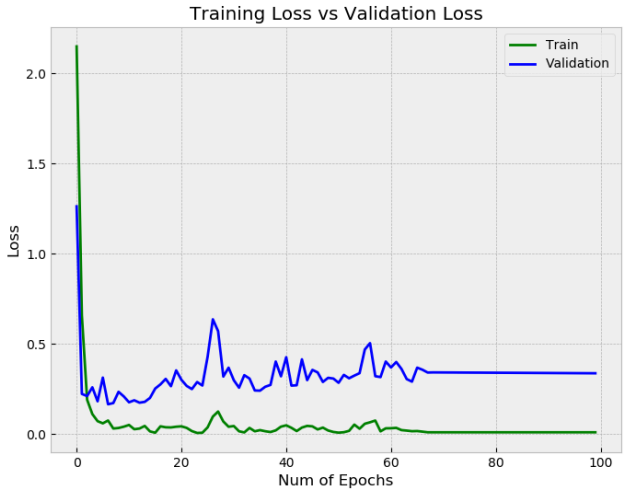

Ilustración 92: Evolución del Loss en la fase de aprendizaje con Ranzato2006 y CJ1

- **Fase de testing**

  En la parte del testing se obtuvo la matriz de confusión que se puede observar en la Ilustración 93, fijándose en los valores de la matriz se puede ver que el modelo empeora ligeramente respecto al último experimento.

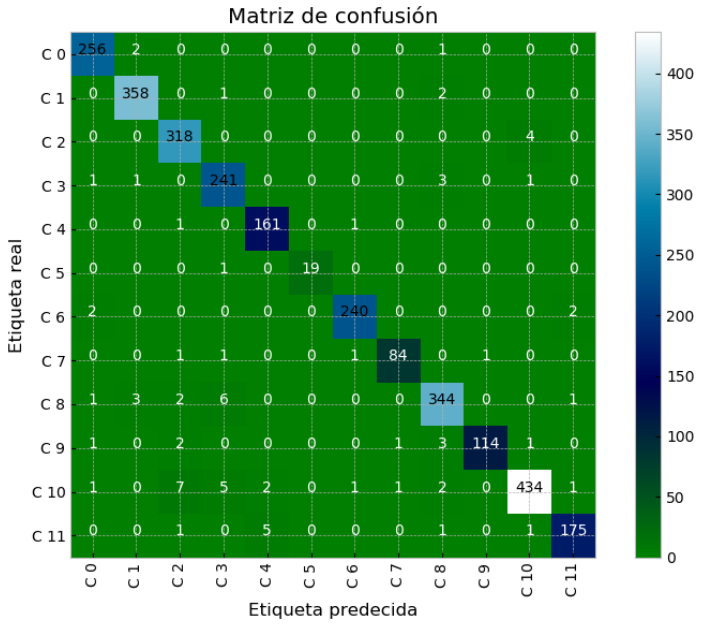

Ilustración 93: Matriz de confusión de la red Ranzato2006 para CJ1

Dicho bajón también se puede ver de manera textual en las métricas obtenidas en la Ilustración 94 y en la Ilustración 95.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	256.0	2555.0	6.0	3.0	0.9968	0.9884	0.9977	0.9771	0.9988	0.0023	0.0116	0.0229	0.9827
C 1	358.0	2453.0	6.0	3.0	0.9968	0.9917	0.9976	0.9835	0.9988	0.0024	0.0083	0.0165	0.9876
C 2	318.0	2484.0	14.0	4.0	0.9936	0.9876	0.9944	0.9578	0.9984	0.0056	0.0124	0.0422	0.9725
C 3	241.0	2559.0	14.0	6.0	0.9929	0.9757	0.9946	0.9451	0.9977	0.0054	0.0243	0.0549	0.9602
C 4	161.0	2650.0	7.0	2.0	0.9968	0.9877	0.9974	0.9583	0.9992	0.0026	0.0123	0.0417	0.9728
C 5	19.0	2800.0	0.0	1.0	0.9996	0.95	1.0	1.0	0.9996	0.0	0.05	0.0	0.9744
C 6	240.0	2573.0	3.0	4.0	0.9975	0.9836	0.9988	0.9877	0.9984	0.0012	0.0164	0.0123	0.9856
C 7	84.0	2730.0	2.0	4.0	0.9979	0.9545	0.9993	0.9767	0.9985	0.0007	0.0455	0.0233	0.9655
C 8	344.0	2451.0	12.0	13.0	0.9911	0.9636	0.9951	0.9663	0.9947	0.0049	0.0364	0.0337	0.9649
C 9	114.0	2697.0	1.0	8.0	0.9968	0.9344	0.9996	0.9913	0.997	0.0004	0.0656	0.0087	0.962
C 10	434.0	2359.0	7.0	20.0	0.9904	0.9559	0.997	0.9841	0.9916	0.003	0.0441	0.0159	0.9698
C 11	175.0	2633.0	4.0	8.0	0.9957	0.9563	0.9985	0.9777	0.997	0.0015	0.0437	0.0223	0.9669

Ilustración 94: Métricas que reflejan el rendimiento del modelo Ranzato2006 para CJ1

ACC	0.973

Ilustración 95: Accuracy de Ranzato2006 para CJ1

Los valores obtenidos para la representación de la curva ROC y el AUC son los mostrados en la Ilustración 96.

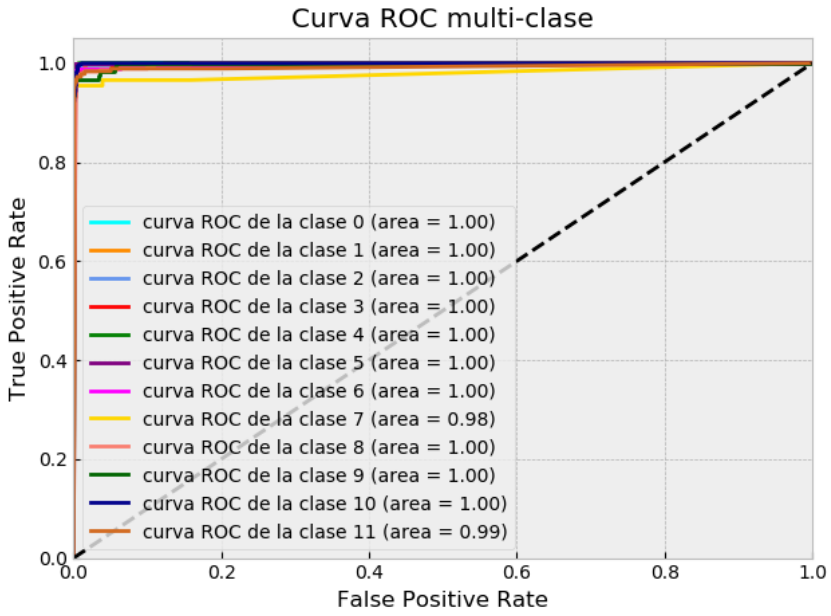

Ilustración 96: Curva ROC y AUC red Ranzato2006 para CJ1

- **Conclusiones**

  Tras analizar los resultados obtenidos con el cambio en el número de los filtros utilizados en las capas de convolución se puede ver que el hecho de aumentar el número de características en el mismo número de capas no presenta ninguna mejora para nuestro conjunto de imágenes. De hecho, lo que agrega es lentitud al proceso de aprendizaje de la red como se pudo ver en la fase en aprendizaje y unos resultados un poco peores obteniendo un ratio de error de **2.69%.**

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  El resultado del entrenamiento realizado para CJ2 se puede ver en la Ilustración 97 y en la Ilustración 98, éste sigue una evolución muy similar a la obtenida con el conjunto CJ1.

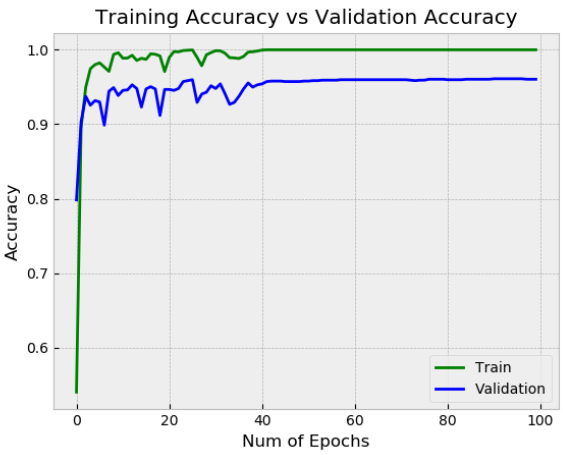

Ilustración 97: Evolución del Accuracy en la fase de aprendizaje con Ranzato2006 y CJ2

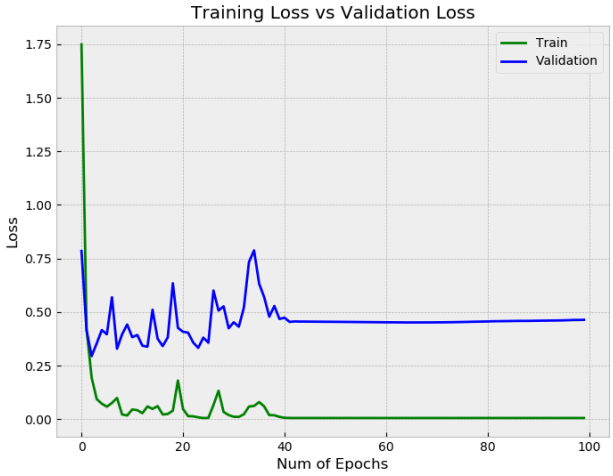

Ilustración 98: Evolución del Loss en la fase de aprendizaje con Ranzato2006 y CJ2

- **Fase de testing**

  El resultado obtenido en forma de matriz de confusión se muestra en la Ilustración 99.

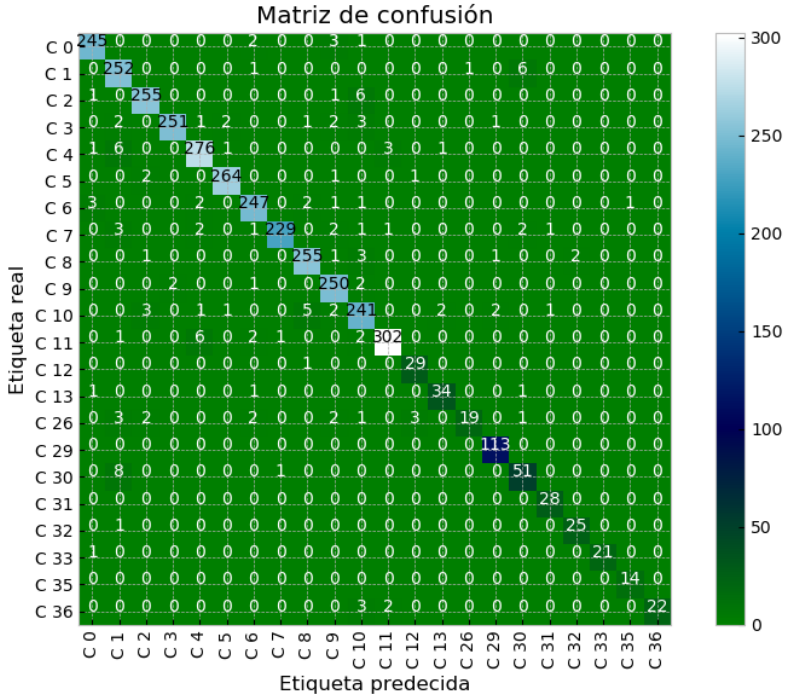

Ilustración 99: Matriz de confusión de la red Ranzato2006 para CJ2

Del mismo modo, las métricas obtenidas por la red son las mostradas en la Ilustración 100 y en la Ilustración 101.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	256.0	2555.0	6.0	3.0	0.9968	0.9884	0.9977	0.9771	0.9988	0.0023	0.0116	0.0229	0.9827
C 1	358.0	2453.0	6.0	3.0	0.9968	0.9917	0.9976	0.9835	0.9988	0.0024	0.0083	0.0165	0.9876
C 2	318.0	2484.0	14.0	4.0	0.9936	0.9876	0.9944	0.9578	0.9984	0.0056	0.0124	0.0422	0.9725
C 3	241.0	2559.0	14.0	6.0	0.9929	0.9757	0.9946	0.9451	0.9977	0.0054	0.0243	0.0549	0.9602
C 4	161.0	2650.0	7.0	2.0	0.9968	0.9877	0.9974	0.9583	0.9992	0.0026	0.0123	0.0417	0.9728
C 5	19.0	2800.0	0.0	1.0	0.9996	0.95	1.0	1.0	0.9996	0.0	0.05	0.0	0.9744
C 6	240.0	2573.0	3.0	4.0	0.9975	0.9836	0.9988	0.9877	0.9984	0.0012	0.0164	0.0123	0.9856
C 7	84.0	2730.0	2.0	4.0	0.9979	0.9545	0.9993	0.9767	0.9985	0.0007	0.0455	0.0233	0.9655
C 8	344.0	2451.0	12.0	13.0	0.9911	0.9636	0.9951	0.9663	0.9947	0.0049	0.0364	0.0337	0.9649
C 9	114.0	2697.0	1.0	8.0	0.9968	0.9344	0.9996	0.9913	0.997	0.0004	0.0656	0.0087	0.962
C 10	434.0	2359.0	7.0	20.0	0.9904	0.9559	0.997	0.9841	0.9916	0.003	0.0441	0.0159	0.9698
C 11	175.0	2633.0	4.0	8.0	0.9957	0.9563	0.9985	0.9777	0.997	0.0015	0.0437	0.0223	0.9669

Ilustración 100: Métricas que reflejan el rendimiento del modelo Ranzato2006 para CJ2

ACC	0.9583

Ilustración 101: Accuracy de Ranzato2006 para CJ2

Por último, la Curva ROC y el AUC obtenidos se muestran en la Ilustración 102.

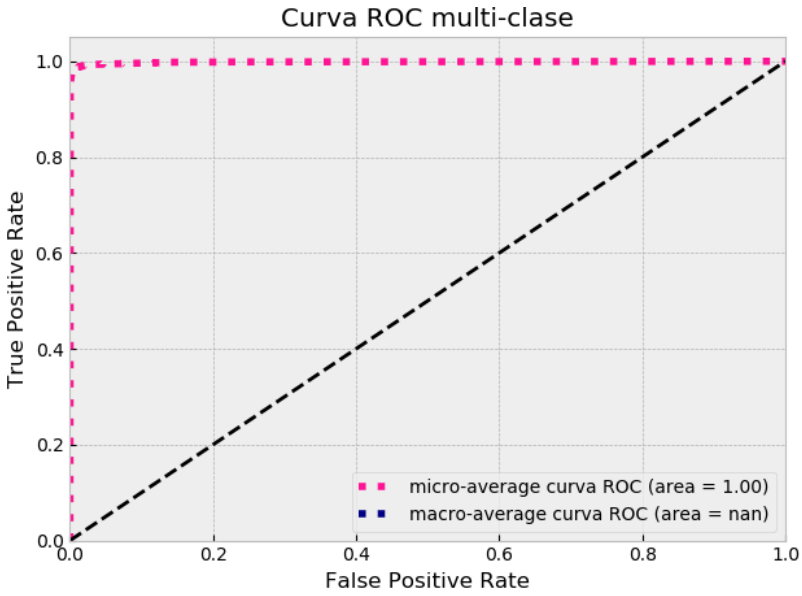

Ilustración 102: Curva ROC y AUC red Ranzato2006 para CJ2

- **Conclusiones**

Al igual que ha pasado con CJ1, agregar más filtros en las capas de convolución no ha mejorado el aprendizaje de la red, sino que la ha ralentizado y emporado su

rendimiento, obteniendo en este caso un ratio de error de **4.17%**, por lo tanto, se seguirá probando con las siguientes redes.

8.1.5 Experimentos con la red Simard2003

La siguiente red a probar será la presentada en el estudio [Simard2003]. En este estudio se hacía hincapié en algunas prácticas para mejorar el rendimiento de los modelos y en él utilizaba una red más sencilla y con menos capas de las que se han utilizado en los experimentos realizados hasta ahora. De modo que, tras ver que aumentar el número de filtros no produjo resultados satisfactorios, se ha decido probar la opción contraria ya que el número de capas y de filtros depende del conjunto de datos.

Arquitectura: [Simard2003]

- Numero de capas: **6 capas**
- Capa convolución
  - Número de filtros: **5, 50**
  - Tamaño kernel: 5
- Capa de pooling
  - Función: Average
  - Tamaño: 2
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  La evolución de la red durante la fase de entrenamiento es la mostrada en la Ilustración 103 y en la Ilustración 104, observando éstas, se puede ver que el sistema aprende de manera muy rápida y que además al final parece producirse el fenómeno denominado overfitting, ya que el valor del Loss parece que aumenta.

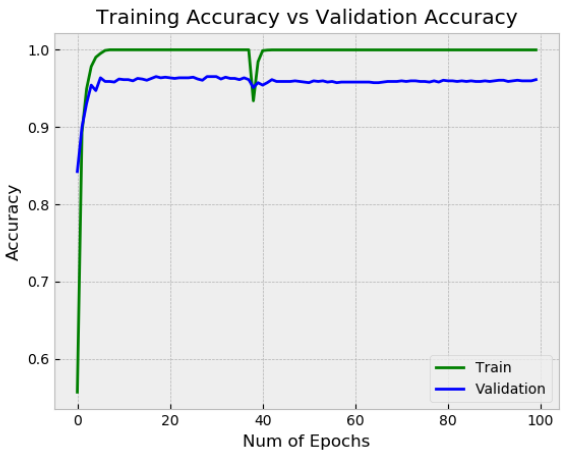

Ilustración 103: Evolución del Accuracy en la fase de aprendizaje con Simard2003 y CJ1

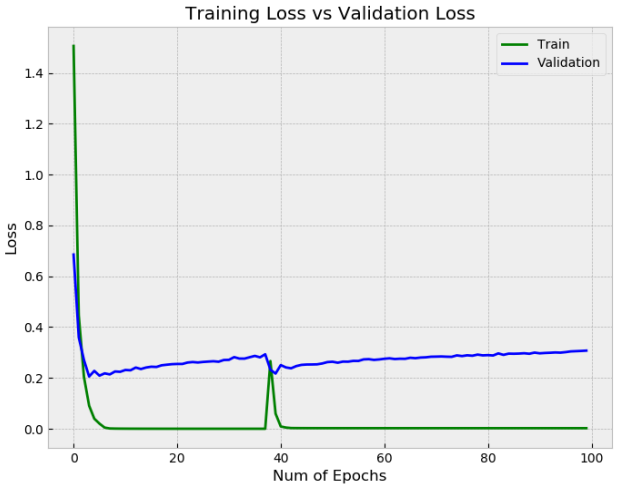

Ilustración 104: Evolución del Loss en la fase de aprendizaje con Simard2003 y CJ1

- **Fase de testing**

  En cuanto a la información recogida en la matriz de confusión de la Ilustración 105, se puede ver que hay algunas clases donde los errores de clasificación son más frecuentes que en alguna de las redes anteriormente probadas.

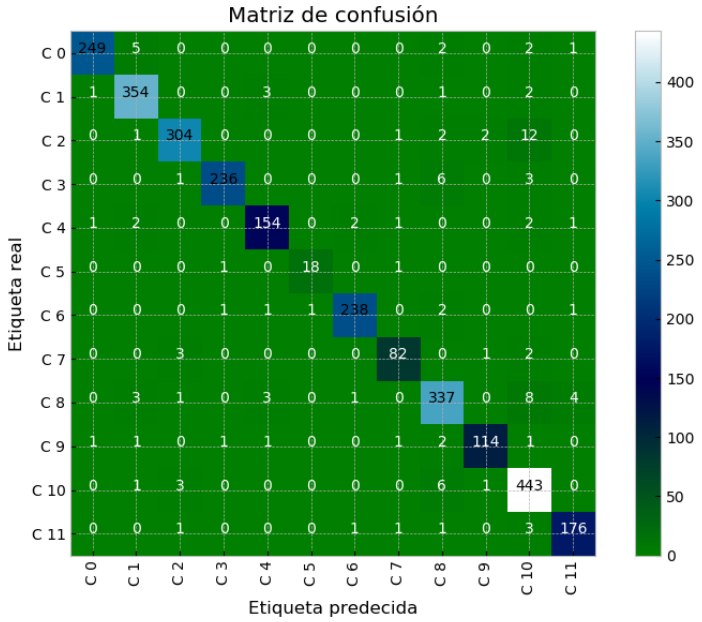

Ilustración 105: Matriz de confusión de la red Simard2003 para CJ1

Por otro lado, en la Ilustración 106 y en la Ilustración 107, las métricas de evaluación dejan ver que la red es algo peor que la anterior, obteniendo un Accuracy del 0.95.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	249.0	2558.0	3.0	10.0	0.9954	0.9614	0.9988	0.9881	0.9961	0.0012	0.0386	0.0119	0.9746
C 1	354.0	2446.0	13.0	7.0	0.9929	0.9806	0.9947	0.9646	0.9971	0.0053	0.0194	0.0354	0.9725
C 2	304.0	2489.0	9.0	18.0	0.9904	0.9441	0.9964	0.9712	0.9928	0.0036	0.0559	0.0288	0.9575
C 3	236.0	2570.0	3.0	11.0	0.995	0.9555	0.9988	0.9874	0.9957	0.0012	0.0445	0.0126	0.9712
C 4	154.0	2649.0	8.0	9.0	0.994	0.9448	0.997	0.9506	0.9966	0.003	0.0552	0.0494	0.9477
C 5	18.0	2799.0	1.0	2.0	0.9989	0.9	0.9996	0.9474	0.9993	0.0004	0.1	0.0526	0.9231
C 6	238.0	2572.0	4.0	6.0	0.9965	0.9754	0.9984	0.9835	0.9977	0.0016	0.0246	0.0165	0.9794
C 7	82.0	2726.0	6.0	6.0	0.9957	0.9318	0.9978	0.9318	0.9978	0.0022	0.0682	0.0682	0.9318
C 8	337.0	2441.0	22.0	20.0	0.9851	0.944	0.9911	0.9387	0.9919	0.0089	0.056	0.0613	0.9413
C 9	114.0	2694.0	4.0	8.0	0.9957	0.9344	0.9985	0.9661	0.997	0.0015	0.0656	0.0339	0.95
C 10	443.0	2331.0	35.0	11.0	0.9837	0.9758	0.9852	0.9268	0.9953	0.0148	0.0242	0.0732	0.9506
C 11	176.0	2630.0	7.0	7.0	0.995	0.9617	0.9973	0.9617	0.9973	0.0027	0.0383	0.0383	0.9617

Ilustración 106: Métricas que reflejan el rendimiento del modelo Simard2003 para CJ1

ACC	0.9592

Ilustración 107: Accuracy de Simard2003 para CJ1

En cambio, lo que respecta a la curva ROC y al AUC mostrados en la Ilustración 108, los valores dejan ver que el modelo es bueno.

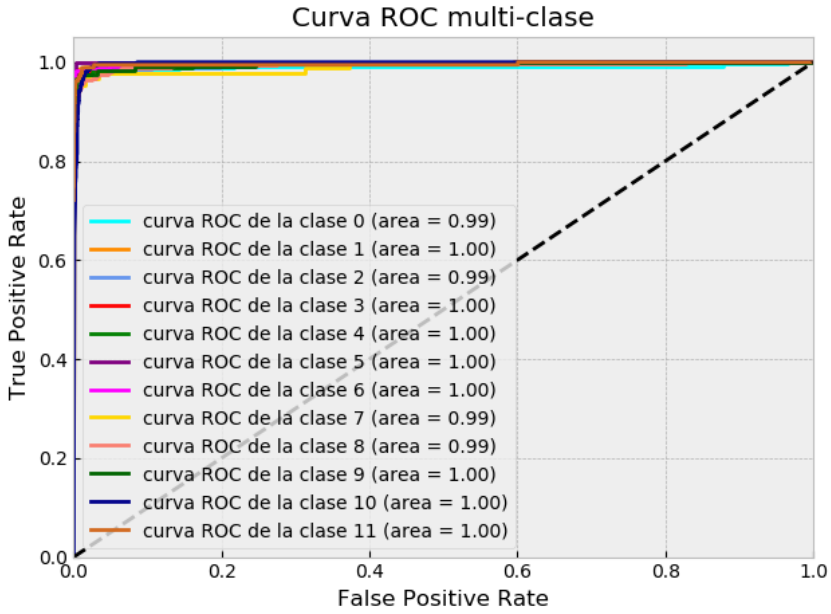

Ilustración 108: Curva ROC y AUC red Simard2003 para CJ1

- **Conclusiones**

  Tras analizar los resultados obtenidos se puede ver que reducir el número de capas de convolución no es una decisión acertada para el conjunto de imágenes que

conforman el dataset, no obstante, el número de filtros utilizados en este caso es mejor que el probado anteriormente, ya que el aprendizaje se realiza de manera más rápida, finalmente se obtuvo un ratio de error de **4.07%.**

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  En el caso de CJ2, como se puede ver en la Ilustración 109 y en la Ilustración 110, la evolución de la red ha desarrollado una tendencia algo diferente respecto a CJ1, los valores parecen ser peores, esto se puede deber a que las imágenes de este dataset están tratadas y sobre todo al ser un conjunto más poblado. No obstante, se ve que al igual que con CJ1 el proceso de aprendizaje se realiza más rápidamente y también se produce el fenómeno de overfitting en este caso mucho más pronunciado.

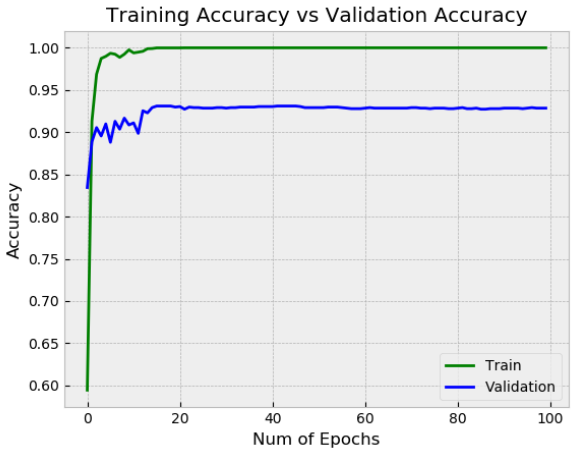

Ilustración 109: Evolución del Accuracy en la fase de aprendizaje con Simard2003 y CJ2

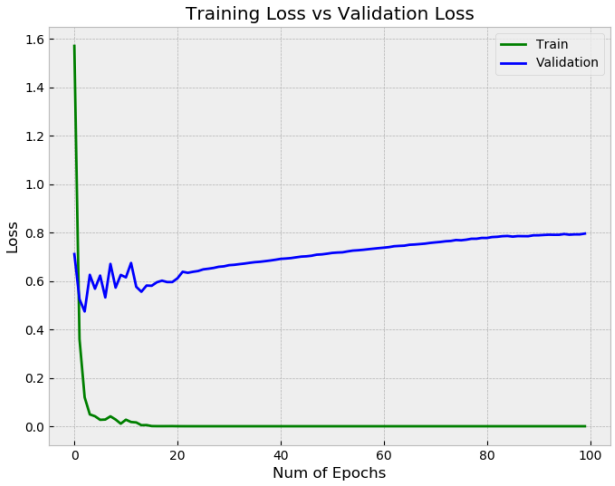

Ilustración 110: Evolución del Loss en la fase de aprendizaje con Simard2003 y CJ2

- **Fase de testing**

  En la fase de testing el resultado obtenido se puede interpretar mediante la matriz de confusión de la Ilustración 111, en ella se ve que hay alguna clase en la que el número de imágenes mal clasificadas es elevado respecto a las demás.

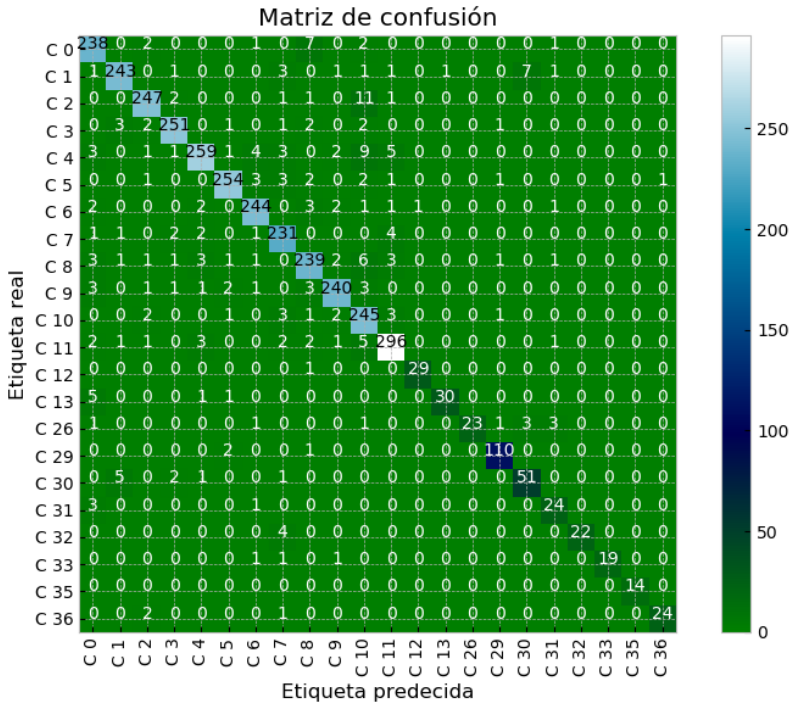

Ilustración 111: Matriz de confusión de la red Simard2003 para CJ2

La situación observaba en la matriz de confusión también se hace visible en los valores de las métricas de la Ilustración 112 y en el valor del Accuracy de la Ilustración 113.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	238.0	3297.0	24.0	13.0	0.9896	0.9482	0.9928	0.9084	0.9961	0.0072	0.0518	0.0916	0.9279
C 1	243.0	3301.0	11.0	17.0	0.9922	0.9346	0.9967	0.9567	0.9949	0.0033	0.0654	0.0433	0.9455
C 2	247.0	3296.0	13.0	16.0	0.9919	0.9392	0.9961	0.95	0.9952	0.0039	0.0608	0.05	0.9446
C 3	251.0	3299.0	10.0	12.0	0.9938	0.9544	0.997	0.9617	0.9964	0.003	0.0456	0.0383	0.958
C 4	259.0	3271.0	13.0	29.0	0.9882	0.8993	0.996	0.9522	0.9912	0.004	0.1007	0.0478	0.925
C 5	254.0	3295.0	9.0	14.0	0.9936	0.9478	0.9973	0.9658	0.9958	0.0027	0.0522	0.0342	0.9567
C 6	244.0	3301.0	14.0	13.0	0.9924	0.9494	0.9958	0.9457	0.9961	0.0042	0.0506	0.0543	0.9476
C 7	231.0	3307.0	23.0	11.0	0.9905	0.9545	0.9931	0.9094	0.9967	0.0069	0.0455	0.0906	0.9315
C 8	239.0	3286.0	23.0	24.0	0.9868	0.9087	0.993	0.9122	0.9927	0.007	0.0913	0.0878	0.9105
C 9	240.0	3306.0	11.0	15.0	0.9927	0.9412	0.9967	0.9562	0.9955	0.0033	0.0588	0.0438	0.9486
C 10	245.0	3271.0	43.0	13.0	0.9843	0.9496	0.987	0.8507	0.996	0.013	0.0504	0.1493	0.8974
C 11	296.0	3239.0	19.0	18.0	0.9896	0.9427	0.9942	0.9397	0.9945	0.0058	0.0573	0.0603	0.9412
C 12	29.0	3541.0	1.0	1.0	0.9994	0.9667	0.9997	0.9667	0.9997	0.0003	0.0333	0.0333	0.9667
C 13	30.0	3534.0	1.0	7.0	0.9978	0.8108	0.9997	0.9677	0.998	0.0003	0.1892	0.0323	0.8824
C 26	23.0	3539.0	0.0	10.0	0.9972	0.697	1.0	1.0	0.9972	0.0	0.303	0.0	0.8214
C 29	110.0	3454.0	5.0	3.0	0.9978	0.9735	0.9986	0.9565	0.9991	0.0014	0.0265	0.0435	0.9649
C 30	51.0	3502.0	10.0	9.0	0.9947	0.85	0.9972	0.8361	0.9974	0.0028	0.15	0.1639	0.843
C 31	24.0	3536.0	8.0	4.0	0.9966	0.8571	0.9977	0.75	0.9989	0.0023	0.1429	0.25	0.8
C 32	22.0	3546.0	0.0	4.0	0.9989	0.8462	1.0	1.0	0.9989	0.0	0.1538	0.0	0.9167
C 33	19.0	3550.0	0.0	3.0	0.9992	0.8636	1.0	1.0	0.9992	0.0	0.1364	0.0	0.9268
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	24.0	3544.0	1.0	3.0	0.9989	0.8889	0.9997	0.96	0.9992	0.0003	0.1111	0.04	0.9231

Ilustración 112: Métricas que reflejan el rendimiento del modelo Simard2003 para CJ2

ACC	0.9331

Ilustración 113: Accuracy de Simard2003 para CJ2

Por último, en la Ilustración 114 se puede observar el resultado global de la red para la Curva ROC y el AUC.

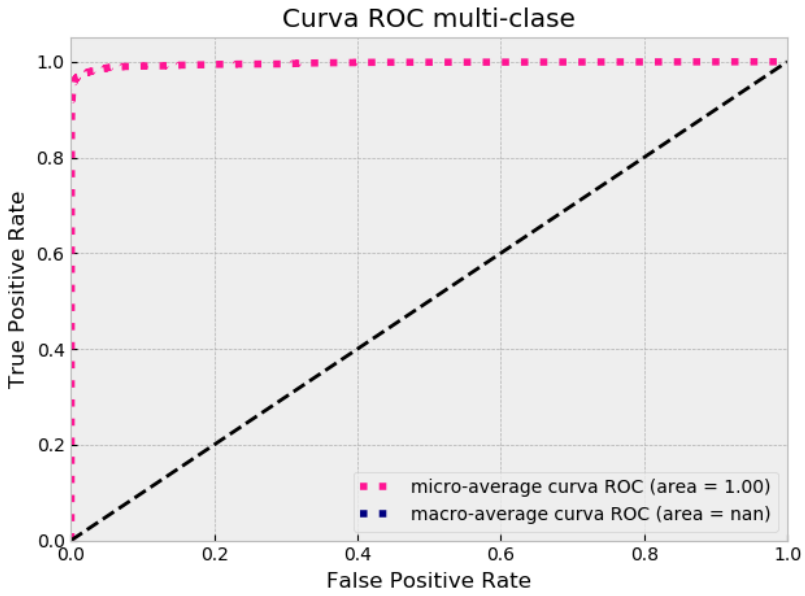

Ilustración 114: Curva ROC y AUC red Simard2003 para CJ2

- **Conclusiones**
  Tras la realización del experimento se puede asegurar que usar dos capas convolucionales para el problema es insuficiente, es decir, tanto para CJ2 como para CJ1, se ha visto que el número de características obtenido no es suficiente, ya que llegado un número de epochs se produce overfitting, por lo tanto, se deben agregar capas. Además, parece que el número de filtros de estas capas debe ascender gradualmente para que el aprendizaje sea fructífero. De este modo, estas pautas se tendrán en cuenta para seguir experimentando con las siguientes arquitecturas. Para este experimento se obtuvo un ratio de error del **6.69%.**

8.1.6 Experimentos con la red Ciresan2011A

Se realiza una experimentación de la red presentada en el estudio [Ciresan2011A], en base al estudio teórico realizado dicha red presenta el mejor rendimiento entre las redes vistas hasta el momento. En cuanto a sus características, cuenta con 3 capas de convolución seguidas de otras de submuestreo, además en este caso utiliza la función max pooling para realizar dicho submuestreo, pudiendo mejorar los resultados frente a la función average utilizada hasta el momento.

Arquitectura: [Ciresan2011A]

- Numero de capas: **8 capas**
- Capa convolución

  - Número de filtros: **20, 60, 100**
  - Tamaño kernel: 5
- Capa de pooling
  - Función: **Max**
  - Tamaño: 2
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  Los resultados de la evolución obtenida en la etapa de aprendizaje se muestran en la Ilustración 115 y en la Ilustración 116. En ellas se puede observar que la red realiza un aprendizaje rápido y lo mantiene constante.

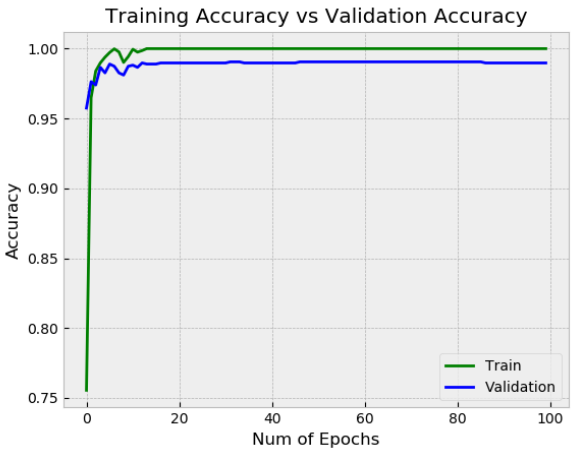

Ilustración 115: Evolución del Accuracy en la fase de aprendizaje con Ciresan2011A y CJ1

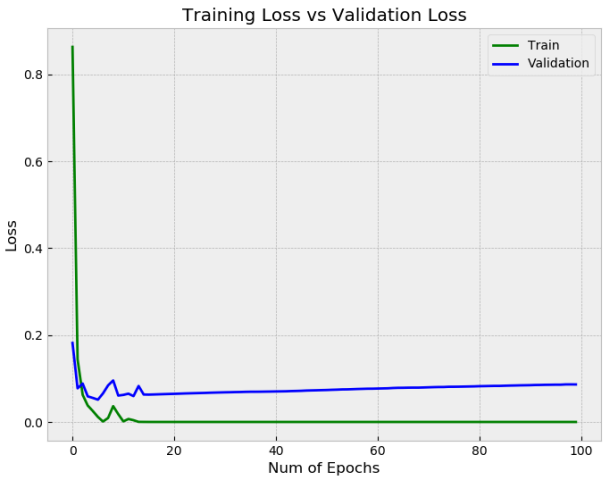

Ilustración 116: Evolución del Loss en la fase de aprendizaje con Ciresan2011A y CJ1

- **Fase de testing**

  En la matriz de confusión de la Ilustración 117, se puede ver reflejado el buen rendimiento de la red.

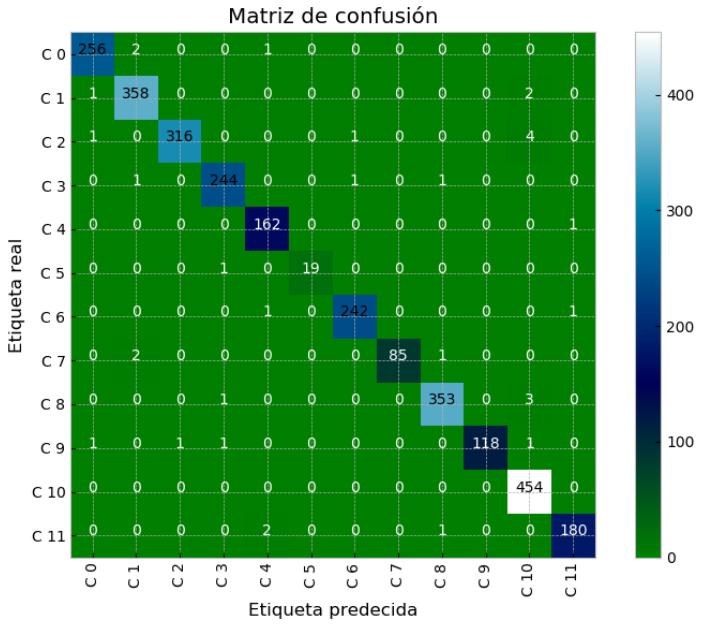

Ilustración 117: Matriz de confusión de la red Ciresan2011A para CJ1

Del mismo modo, las métricas de la Ilustración 118 y la Ilustración 119 mantienen de manera numérica los buenos resultados.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	256.0	2558.0	3.0	3.0	0.9979	0.9884	0.9988	0.9884	0.9988	0.0012	0.0116	0.0116	0.9884
C 1	358.0	2454.0	5.0	3.0	0.9972	0.9917	0.998	0.9862	0.9988	0.002	0.0083	0.0138	0.989
C 2	316.0	2497.0	1.0	6.0	0.9975	0.9814	0.9996	0.9968	0.9976	0.0004	0.0186	0.0032	0.989
C 3	244.0	2570.0	3.0	3.0	0.9979	0.9879	0.9988	0.9879	0.9988	0.0012	0.0121	0.0121	0.9879
C 4	162.0	2653.0	4.0	1.0	0.9982	0.9939	0.9985	0.9759	0.9996	0.0015	0.0061	0.0241	0.9848
C 5	19.0	2800.0	0.0	1.0	0.9996	0.95	1.0	1.0	0.9996	0.0	0.05	0.0	0.9744
C 6	242.0	2574.0	2.0	2.0	0.9986	0.9918	0.9992	0.9918	0.9992	0.0008	0.0082	0.0082	0.9918
C 7	85.0	2732.0	0.0	3.0	0.9989	0.9659	1.0	1.0	0.9989	0.0	0.0341	0.0	0.9827
C 8	353.0	2460.0	3.0	4.0	0.9975	0.9888	0.9988	0.9916	0.9984	0.0012	0.0112	0.0084	0.9902
C 9	118.0	2698.0	0.0	4.0	0.9986	0.9672	1.0	1.0	0.9985	0.0	0.0328	0.0	0.9833
C 10	454.0	2356.0	10.0	0.0	0.9965	1.0	0.9958	0.9784	1.0	0.0042	0.0	0.0216	0.9891
C 11	180.0	2635.0	2.0	3.0	0.9982	0.9836	0.9992	0.989	0.9989	0.0008	0.0164	0.011	0.9863

Ilustración 118: Métricas que reflejan el rendimiento del modelo Ciresan2011A para CJ1

ACC	0.9883

Ilustración 119: Accuracy de Ciresan2011A para CJ1

Por último, según la Ilustración 120 la Curva ROC y el AUC de la red son inmejorables.

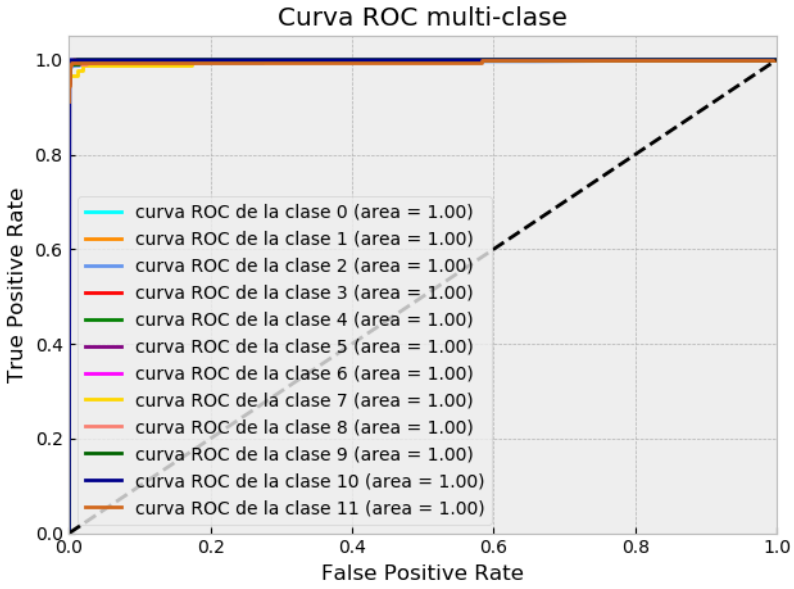

Ilustración 120: Curva ROC y AUC red Ciresan2011A para CJ1

- **Conclusiones**

  Los resultados obtenidos por la red arrojan un buen funcionamiento de la combinación de características, pudiendo ser una buena red en la que profundizar de cara a una mejora mayor del rendimiento, debido a que el ratio de error obtenido es de un **1.17%.**

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  Para CJ2 la evolución del entrenamiento de la red se muestra en la Ilustración 121 y en la Ilustración 122, en ellas se ve una red que proporciona buenos resultados.

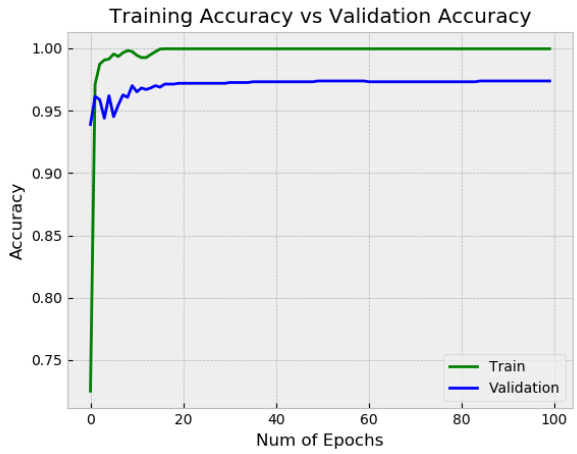

Ilustración 121: Evolución del Accuracy en la fase de aprendizaje con Ciresan2011A y CJ2

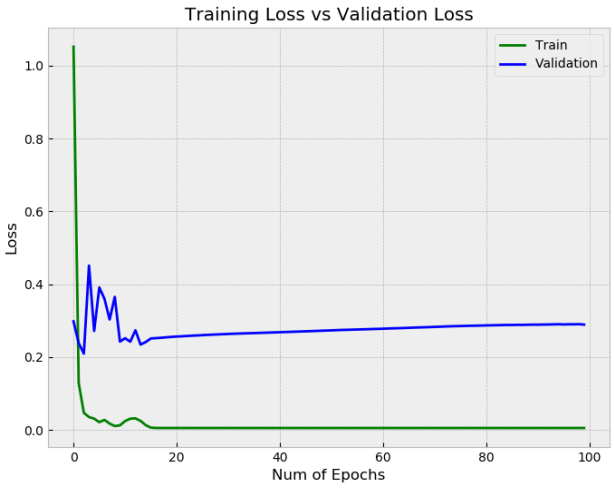

Ilustración 122: Evolución del Loss en la fase de aprendizaje con Ciresan2011A y CJ2

- **Fase de testing**

  Observando la Ilustración 123, la matriz de confusión refleja errores de clasificación mayormente en los dígitos.

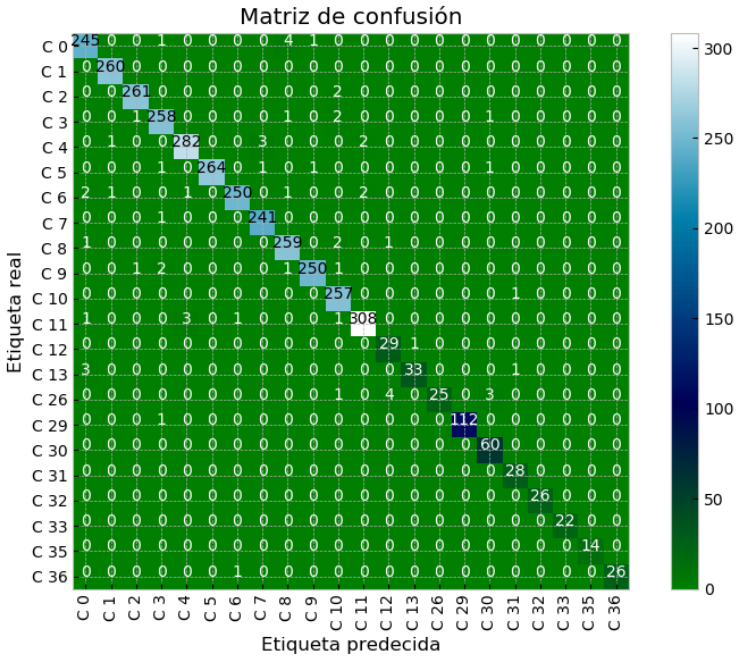

Ilustración 123: Matriz de confusión de la red Ciresan2011A para CJ2

En cuanto a los valores numéricos, en la Ilustración 124 y en la Ilustración 125, se puede ver que no existe una diferencia abultada respecto a CJ1, obteniendo buenos resultados en ambos.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	245.0	3314.0	7.0	6.0	0.9964	0.9761	0.9979	0.9722	0.9982	0.0021	0.0239	0.0278	0.9742
C 1	260.0	3310.0	2.0	0.0	0.9994	1.0	0.9994	0.9924	1.0	0.0006	0.0	0.0076	0.9962
C 2	261.0	3307.0	2.0	2.0	0.9989	0.9924	0.9994	0.9924	0.9994	0.0006	0.0076	0.0076	0.9924
C 3	258.0	3303.0	6.0	5.0	0.9969	0.981	0.9982	0.9773	0.9985	0.0018	0.019	0.0227	0.9791
C 4	282.0	3280.0	4.0	6.0	0.9972	0.9792	0.9988	0.986	0.9982	0.0012	0.0208	0.014	0.9826
C 5	264.0	3304.0	0.0	4.0	0.9989	0.9851	1.0	1.0	0.9988	0.0	0.0149	0.0	0.9925
C 6	250.0	3313.0	2.0	7.0	0.9975	0.9728	0.9994	0.9921	0.9979	0.0006	0.0272	0.0079	0.9823
C 7	241.0	3326.0	4.0	1.0	0.9986	0.9959	0.9988	0.9837	0.9997	0.0012	0.0041	0.0163	0.9897
C 8	259.0	3302.0	7.0	4.0	0.9969	0.9848	0.9979	0.9737	0.9988	0.0021	0.0152	0.0263	0.9792
C 9	250.0	3315.0	2.0	5.0	0.998	0.9804	0.9994	0.9921	0.9985	0.0006	0.0196	0.0079	0.9862
C 10	257.0	3305.0	9.0	1.0	0.9972	0.9961	0.9973	0.9662	0.9997	0.0027	0.0039	0.0338	0.9809
C 11	308.0	3254.0	4.0	6.0	0.9972	0.9809	0.9988	0.9872	0.9982	0.0012	0.0191	0.0128	0.984
C 12	29.0	3537.0	5.0	1.0	0.9983	0.9667	0.9986	0.8529	0.9997	0.0014	0.0333	0.1471	0.9062
C 13	33.0	3534.0	1.0	4.0	0.9986	0.8919	0.9997	0.9706	0.9989	0.0003	0.1081	0.0294	0.9296
C 26	25.0	3539.0	0.0	8.0	0.9978	0.7576	1.0	1.0	0.9977	0.0	0.2424	0.0	0.8621
C 29	112.0	3459.0	0.0	1.0	0.9997	0.9912	1.0	1.0	0.9997	0.0	0.0088	0.0	0.9956
C 30	60.0	3507.0	5.0	0.0	0.9986	1.0	0.9986	0.9231	1.0	0.0014	0.0	0.0769	0.96
C 31	28.0	3542.0	2.0	0.0	0.9994	1.0	0.9994	0.9333	1.0	0.0006	0.0	0.0667	0.9655
C 32	26.0	3546.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	26.0	3545.0	0.0	1.0	0.9997	0.963	1.0	1.0	0.9997	0.0	0.037	0.0	0.9811

Ilustración 124: Métricas que reflejan el rendimiento del modelo Ciresan2011A para CJ2

ACC	0.9826

Ilustración 125: Accuracy de Ciresan2011A para CJ2

Finalmente, la Curva ROC y el AUC de la Ilustración 126, respaldan dicho buen funcionamiento.

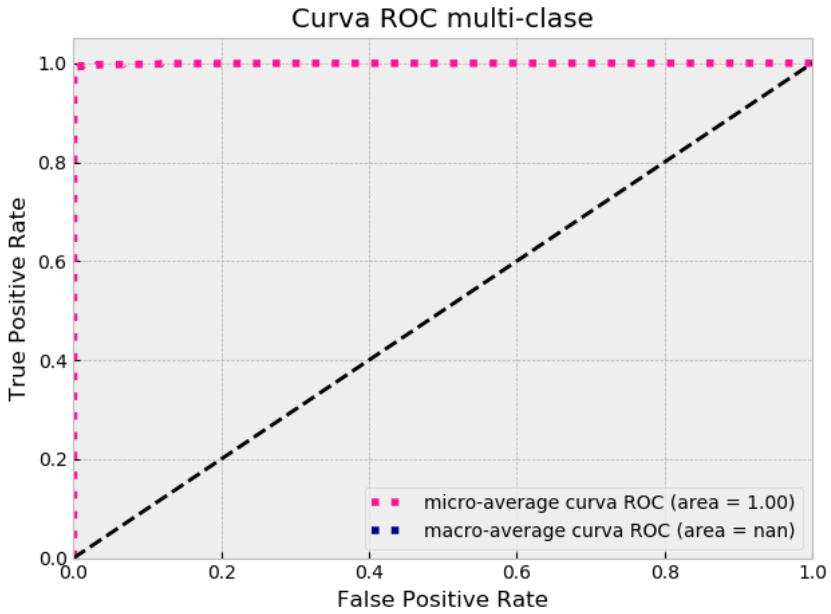

Ilustración 126: Curva ROC y AUC red Ciresan2011A para CJ2

- **Conclusiones**
  Tratándose de un experimento con la red original del estudio sin realizar cambios, el resultado refleja un buen funcionamiento en el cual se podría profundizar de cara a una solución, ya que le ratio de error obtenido es de **1.73%.**

8.1.7 Experimentos con la red Ciresan2011B-Meier2011-Ciresan2012

Tras el estudio anterior el mismo autor realizó dos estudios posteriores [Ciresan2011B] y [Ciresan2012], además colaboro en el estudio [Meier2011], utilizando en todos estos estudios una arquitectura común, difiriendo únicamente en cuanto a la forma de experimentar con ella. Debido a los buenos resultados obtenidos con su trabajo anterior a continuación se realizará una prueba de dicha arquitectura con el problema presentado, ya que, aunque el número de capas disminuye, en las capas de submuestreo experimenta con parámetros diferentes.

Arquitectura: [Ciresan2011B]

- Numero de capas: **6 capas**

- Capa convolución
  - Número de filtros: **20, 40**
  - Tamaño kernel: **4**
- Capa de pooling
  - Función: **Max**
  - Tamaño: **2, 3**
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  Los resultados obtenidos durante el entrenamiento de la red se pueden ver en la Ilustración 127 y en la Ilustración 128. En ellas se ve que la tendencia es muy similar a la red anterior, no obstante, los resultados empeoran.

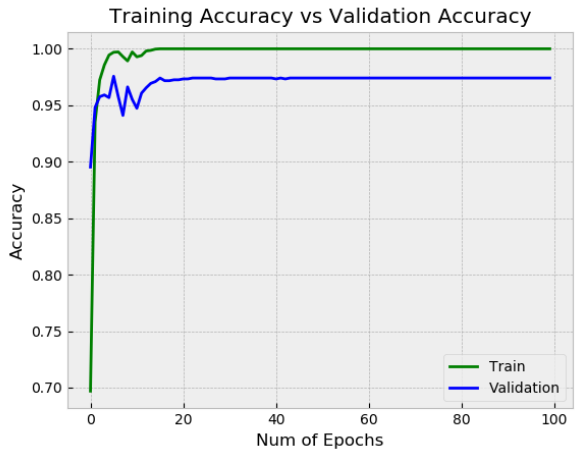

Ilustración 127: Evolución del Accuracy en la fase de aprendizaje con Ciresan2011B y CJ1

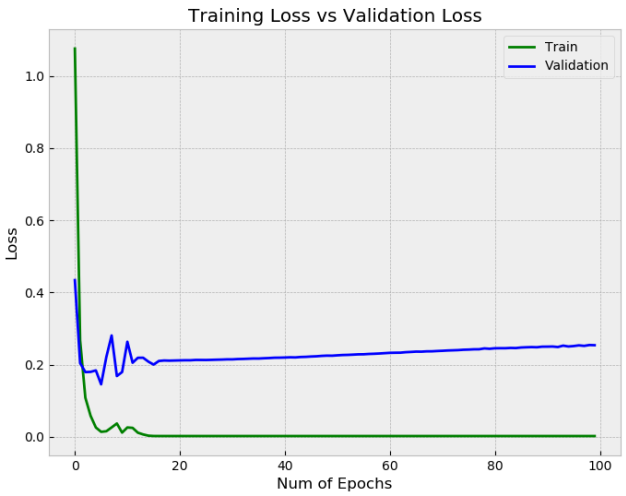

Ilustración 128: Evolución del Loss en la fase de aprendizaje con Ciresan2011B y CJ1

- **Fase de testing**

  Como se ha visto durante el entrenamiento en la matriz de la Ilustración 129, también se refleja un mayor número de errores de clasificación respecto a la red previa.

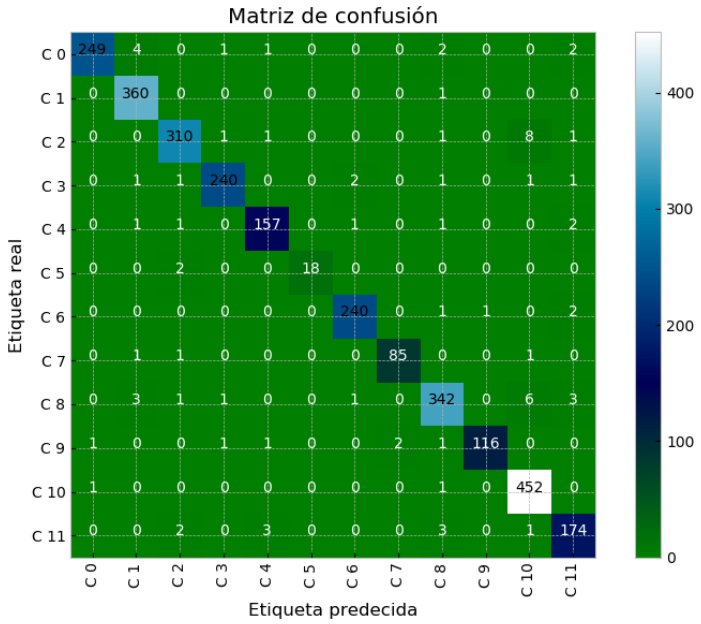

Ilustración 129: Matriz de confusión de la red Ciresan2011B para CJ1

Numéricamente el deterioro de los resultados se puede ver en la Ilustración 130 y en la Ilustración 131.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	249.0	2559.0	2.0	10.0	0.9957	0.9614	0.9992	0.992	0.9961	0.0008	0.0386	0.008	0.9765
C 1	360.0	2449.0	10.0	1.0	0.9961	0.9972	0.9959	0.973	0.9996	0.0041	0.0028	0.027	0.985
C 2	310.0	2490.0	8.0	12.0	0.9929	0.9627	0.9968	0.9748	0.9952	0.0032	0.0373	0.0252	0.9687
C 3	240.0	2569.0	4.0	7.0	0.9961	0.9717	0.9984	0.9836	0.9973	0.0016	0.0283	0.0164	0.9776
C 4	157.0	2651.0	6.0	6.0	0.9957	0.9632	0.9977	0.9632	0.9977	0.0023	0.0368	0.0368	0.9632
C 5	18.0	2800.0	0.0	2.0	0.9993	0.9	1.0	1.0	0.9993	0.0	0.1	0.0	0.9474
C 6	240.0	2572.0	4.0	4.0	0.9972	0.9836	0.9984	0.9836	0.9984	0.0016	0.0164	0.0164	0.9836
C 7	85.0	2730.0	2.0	3.0	0.9982	0.9659	0.9993	0.977	0.9989	0.0007	0.0341	0.023	0.9714
C 8	342.0	2451.0	12.0	15.0	0.9904	0.958	0.9951	0.9661	0.9939	0.0049	0.042	0.0339	0.962
C 9	116.0	2697.0	1.0	6.0	0.9975	0.9508	0.9996	0.9915	0.9978	0.0004	0.0492	0.0085	0.9707
C 10	452.0	2349.0	17.0	2.0	0.9933	0.9956	0.9928	0.9638	0.9991	0.0072	0.0044	0.0362	0.9794
C 11	174.0	2626.0	11.0	9.0	0.9929	0.9508	0.9958	0.9405	0.9966	0.0042	0.0492	0.0595	0.9457

Ilustración 130: Métricas que reflejan el rendimiento del modelo Ciresan2011B para CJ1

ACC	0.9727

Ilustración 131: Accuracy de Ciresan2011B para CJ1

Por último, en la Curva ROC y AUC de la Ilustración 132, se ve como en algunas clases ya no se consigue el área total.

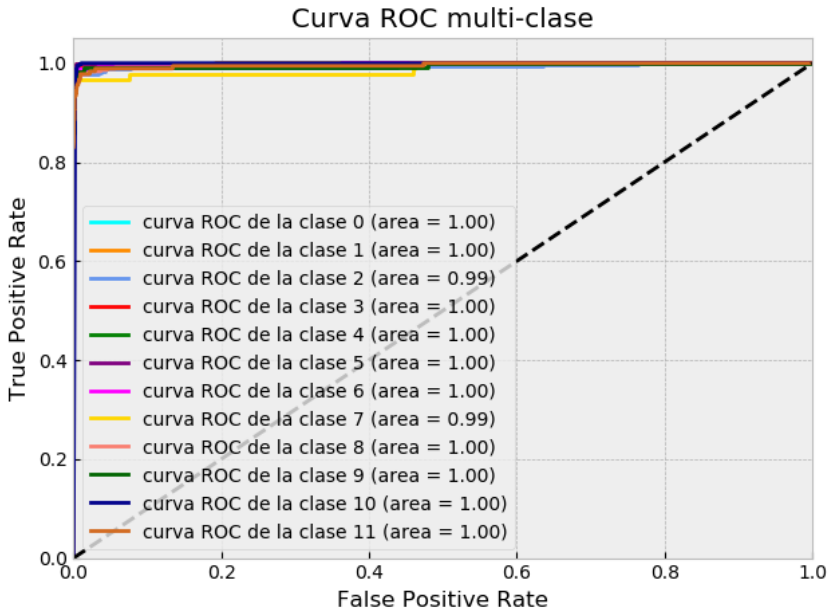

Ilustración 132: Curva ROC y AUC red Ciresan2011B para CJ1

- **Conclusiones**
  Este experimento ratifica que el número de capas de convolución es importante para el conjunto de imágenes utilizado, de modo que, siendo un parámetro a aumentar en futuros experimentos y no al contrario, ya que el ratio de error ha incrementado hasta un valor de **2.73%**. Por lo tanto, viendo que esta red no mejora a la anterior se omitirá la explicación para el CJ2, debido a que se ha observado el mismo funcionamiento con un ratio de error de **5.12**%.

8.1.8 Experimentos con la red Chen2015

Debido a su fecha de realización, el estudio [Chen2015] pretende la mejora de los sistemas de reconocimiento basados en CNN existentes, y además pretende que dicho sistema sea sencillo a modo de una "caja negra". En este experimento probaremos la red propuesta, en la cual se juega con el número de filtros en las capas de convolución y el tamaño del kernel de pooling de este tipo de capas.

Arquitectura: [Chen2015]

- Numero de capas: **8 capas**
- Capa convolución
  - Número de filtros: **32, 64, 128**
  - Tamaño kernel: **5, 3, 3**
- Capa de pooling
  - Función: **Max**
  - Tamaño: **3**

- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  La fase de entrenamiento deja como resultado la evolución de la Ilustración 133 y la Ilustración 134, observando una dinámica que augura buenos resultados.

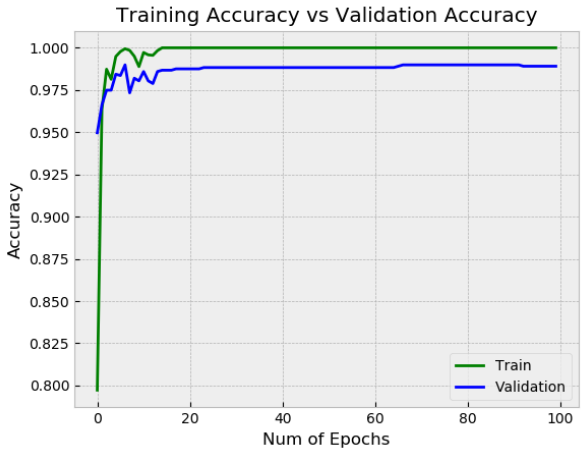

Ilustración 133: Evolución del Accuracy en la fase de aprendizaje con Chen2015 y CJ1

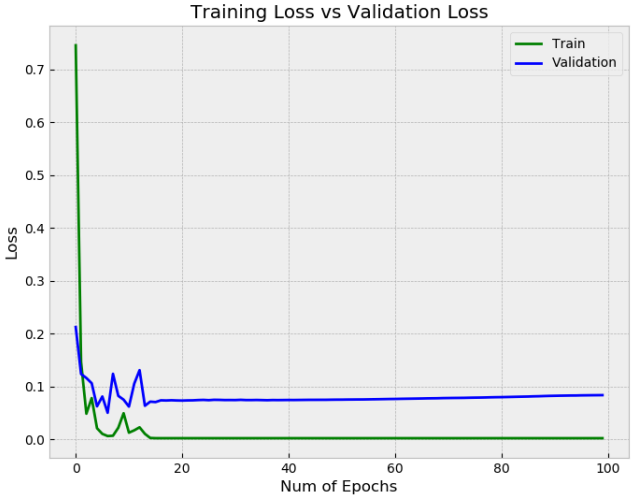

Ilustración 134: Evolución del Loss en la fase de aprendizaje con Chen2015 y CJ1

- **Fase de testing**

  La matriz de la Ilustración 135**,** refleja un buen funcionamiento teniendo un único punto de confusión.

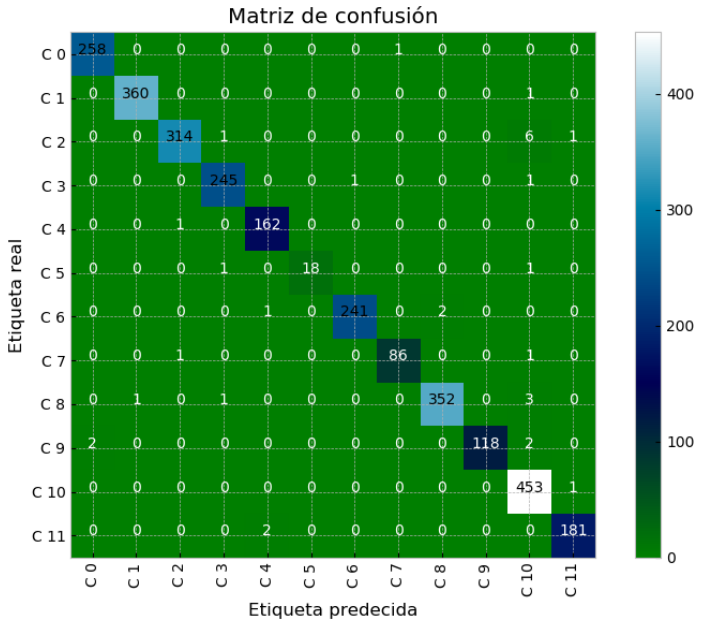

Ilustración 135: Matriz de confusión de la red Chen2015 para CJ1

De este modo, los resultados numéricos presentados en la Ilustración 136 y en la Ilustración 137, ratifican el buen rendimiento de la red.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	258.0	2559.0	2.0	1.0	0.9989	0.9961	0.9992	0.9923	0.9996	0.0008	0.0039	0.0077	0.9942
C 1	360.0	2458.0	1.0	1.0	0.9993	0.9972	0.9996	0.9972	0.9996	0.0004	0.0028	0.0028	0.9972
C 2	314.0	2496.0	2.0	8.0	0.9965	0.9752	0.9992	0.9937	0.9968	0.0008	0.0248	0.0063	0.9843
C 3	245.0	2570.0	3.0	2.0	0.9982	0.9919	0.9988	0.9879	0.9992	0.0012	0.0081	0.0121	0.9899
C 4	162.0	2654.0	3.0	1.0	0.9986	0.9939	0.9989	0.9818	0.9996	0.0011	0.0061	0.0182	0.9878
C 5	18.0	2800.0	0.0	2.0	0.9993	0.9	1.0	1.0	0.9993	0.0	0.1	0.0	0.9474
C 6	241.0	2575.0	1.0	3.0	0.9986	0.9877	0.9996	0.9959	0.9988	0.0004	0.0123	0.0041	0.9918
C 7	86.0	2731.0	1.0	2.0	0.9989	0.9773	0.9996	0.9885	0.9993	0.0004	0.0227	0.0115	0.9829
C 8	352.0	2461.0	2.0	5.0	0.9975	0.986	0.9992	0.9944	0.998	0.0008	0.014	0.0056	0.9902
C 9	118.0	2698.0	0.0	4.0	0.9986	0.9672	1.0	1.0	0.9985	0.0	0.0328	0.0	0.9833
C 10	453.0	2351.0	15.0	1.0	0.9943	0.9978	0.9937	0.9679	0.9996	0.0063	0.0022	0.0321	0.9826
C 11	181.0	2635.0	2.0	2.0	0.9986	0.9891	0.9992	0.9891	0.9992	0.0008	0.0109	0.0109	0.9891

Ilustración 136: Métricas que reflejan el rendimiento del modelo Chen2015 para CJ1

ACC	0.9887

Ilustración 137: Accuracy de Chen2015 para CJ1

Por último, el valor de la Curva ROC y el AUC mostrados en la Ilustración 138, son acordes con el desempeño de la red.

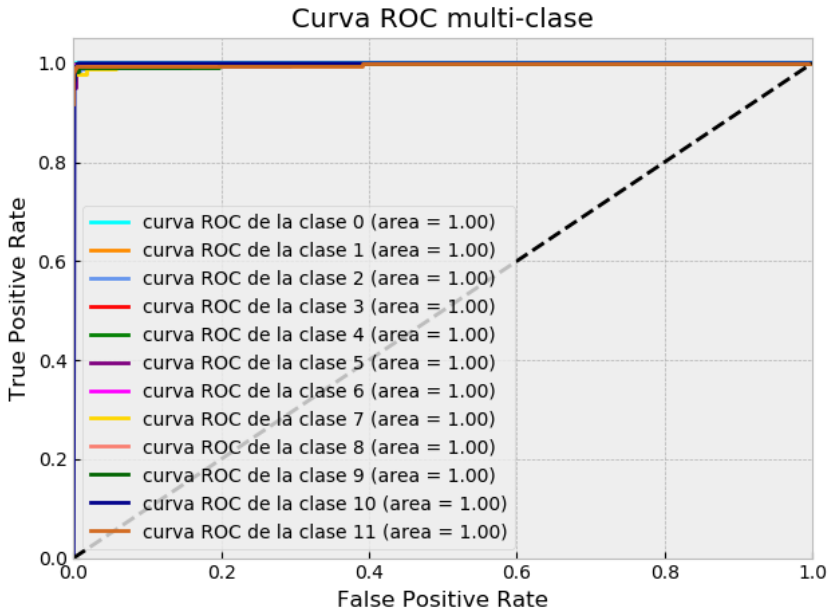

Ilustración 138: Curva ROC y AUC red Chen2015 para CJ1

- **Conclusiones**
  El funcionamiento de la red señala el camino que se debe seguir para mejorar los resultados, las características de la red se asemejan a la probada en Ciresan2011A, obteniendo en este caso un error de ratio de **1.13%.**

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**
  La fase de entrenamiento y validación reflejada en la Ilustración 139 y en la Ilustración 140, sigue la línea vista en CJ1.

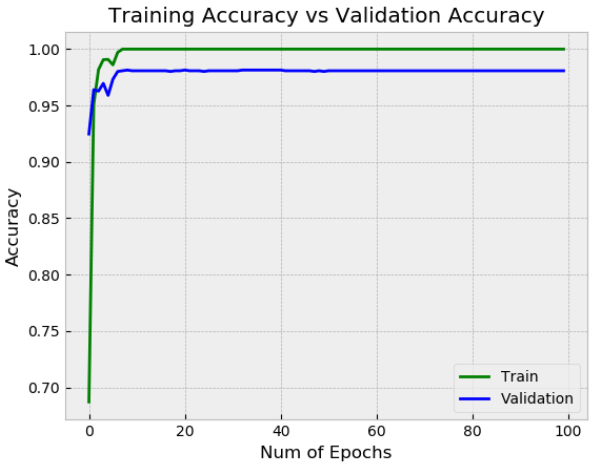

Ilustración 139: Evolución del Accuracy en la fase de aprendizaje con Chen2015 y CJ2

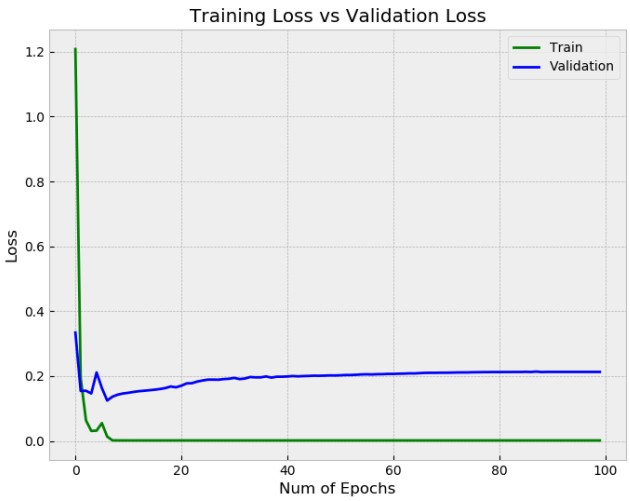

Ilustración 140: Evolución del Loss en la fase de aprendizaje con Chen2015 y CJ2

- **Fase de testing**

  En la matriz de la Ilustración 141, se destaca el mismo punto de confusión que en CJ1 para las imágenes en las que no se encuentran caracteres**.**

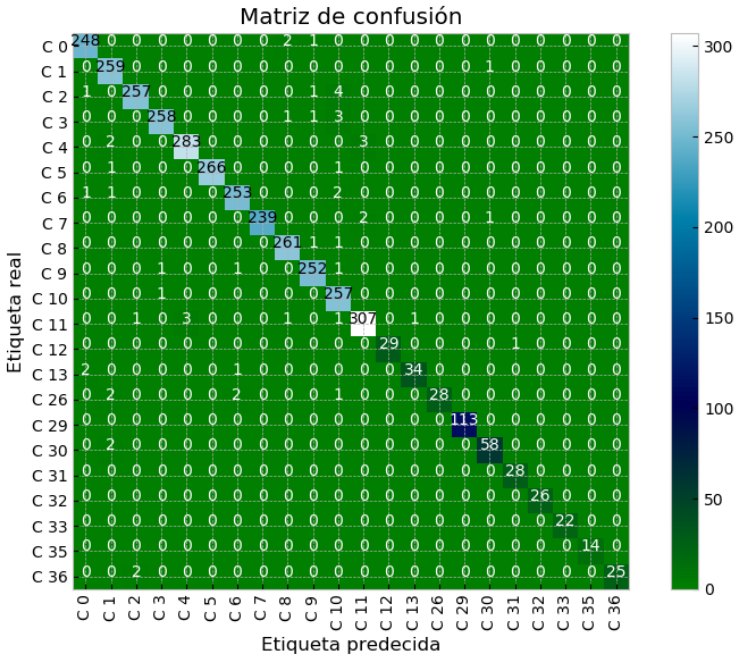

Ilustración 141: Matriz de confusión de la red Chen2015 para CJ2

El valor de las métricas presente en la Ilustración 142 y en la Ilustración 143, reflejan que es la mejor red probada hasta el momento.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	248.0	3317.0	4.0	3.0	0.998	0.988	0.9988	0.9841	0.9991	0.0012	0.012	0.0159	0.9861
C 1	259.0	3304.0	8.0	1.0	0.9975	0.9962	0.9976	0.97	0.9997	0.0024	0.0038	0.03	0.9829
C 2	257.0	3306.0	3.0	6.0	0.9975	0.9772	0.9991	0.9885	0.9982	0.0009	0.0228	0.0115	0.9828
C 3	258.0	3307.0	2.0	5.0	0.998	0.981	0.9994	0.9923	0.9985	0.0006	0.019	0.0077	0.9866
C 4	283.0	3281.0	3.0	5.0	0.9978	0.9826	0.9991	0.9895	0.9985	0.0009	0.0174	0.0105	0.9861
C 5	266.0	3304.0	0.0	2.0	0.9994	0.9925	1.0	1.0	0.9994	0.0	0.0075	0.0	0.9963
C 6	253.0	3311.0	4.0	4.0	0.9978	0.9844	0.9988	0.9844	0.9988	0.0012	0.0156	0.0156	0.9844
C 7	239.0	3330.0	0.0	3.0	0.9992	0.9876	1.0	1.0	0.9991	0.0	0.0124	0.0	0.9938
C 8	261.0	3305.0	4.0	2.0	0.9983	0.9924	0.9988	0.9849	0.9994	0.0012	0.0076	0.0151	0.9886
C 9	252.0	3313.0	4.0	3.0	0.998	0.9882	0.9988	0.9844	0.9991	0.0012	0.0118	0.0156	0.9863
C 10	257.0	3300.0	14.0	1.0	0.9958	0.9961	0.9958	0.9483	0.9997	0.0042	0.0039	0.0517	0.9716
C 11	307.0	3253.0	5.0	7.0	0.9966	0.9777	0.9985	0.984	0.9979	0.0015	0.0223	0.016	0.9808
C 12	29.0	3542.0	0.0	1.0	0.9997	0.9667	1.0	1.0	0.9997	0.0	0.0333	0.0	0.9831
C 13	34.0	3534.0	1.0	3.0	0.9989	0.9189	0.9997	0.9714	0.9992	0.0003	0.0811	0.0286	0.9444
C 26	28.0	3539.0	0.0	5.0	0.9986	0.8485	1.0	1.0	0.9986	0.0	0.1515	0.0	0.918
C 29	113.0	3459.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 30	58.0	3510.0	2.0	2.0	0.9989	0.9667	0.9994	0.9667	0.9994	0.0006	0.0333	0.0333	0.9667
C 31	28.0	3543.0	1.0	0.0	0.9997	1.0	0.9997	0.9655	1.0	0.0003	0.0	0.0345	0.9825
C 32	26.0	3546.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	25.0	3545.0	0.0	2.0	0.9994	0.9259	1.0	1.0	0.9994	0.0	0.0741	0.0	0.9615

Ilustración 142: Métricas que reflejan el rendimiento del modelo Chen2015 para CJ2

ACC	0.9846

Ilustración 143: Accuracy de Chen2015 para CJ2

De igual manera, la Curva ROC y el AUC obtenidos son buenos como se puede ver en la Ilustración 144.

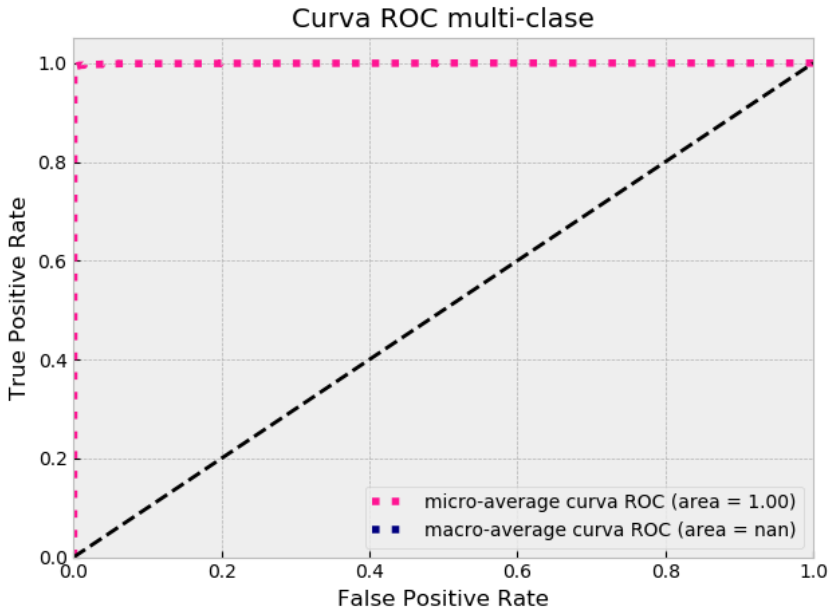

Ilustración 144: Curva ROC y AUC red Chen2015 para CJ2

- **Conclusiones**

  El resultado obtenido con la red refleja que hasta el momento es la mejor red probada, esto se entiende claramente teniendo en cuenta las características y muestra el tipo de mejoras a realizar para poder mejora el **1.53%** de ratio de error obtenido en este caso**.**

8.1.9 Experimentos con la red Alsharif2013

Hasta ahora las redes probadas iban dirigidas a reconocer únicamente dígitos, en el estudio [Alsharif2013] el objetivo se amplía al reconocimiento de todo el alfabeto, de este modo, se podrá analizar el rendimiento teniendo en cuenta que el problema a resolver es de la misma magnitud. A priori, realiza una experimentación con diferente número de filtros en la convolución, pudiendo ser este factor una mejoría.

Arquitectura: [Alsharif2013]

- Numero de capas: **8 capas**
- Capa convolución
  - Número de filtros: **48, 128, 128**
  - Tamaño kernel: **4, 8, 5**
- Capa de pooling
  - Función: **Max**
  - Tamaño: 4
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**
  Los resultados del entrenamiento y validación de la red son visibles en la Ilustración 145 y en la Ilustración 146, éstos presentan un buen rendimiento que hay que analizar.

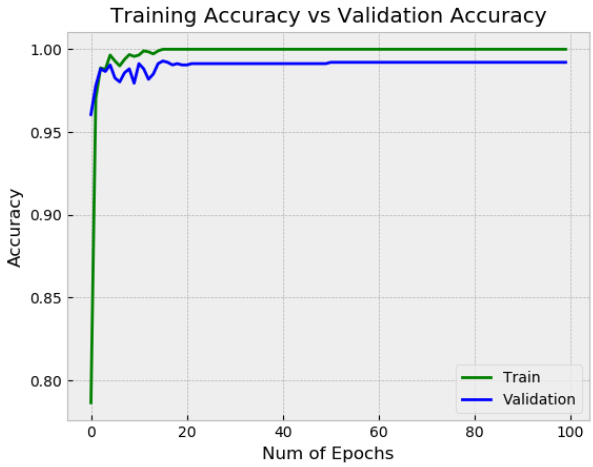

Ilustración 145: Evolución del Accuracy en la fase de aprendizaje con Alsharif2013 y CJ1

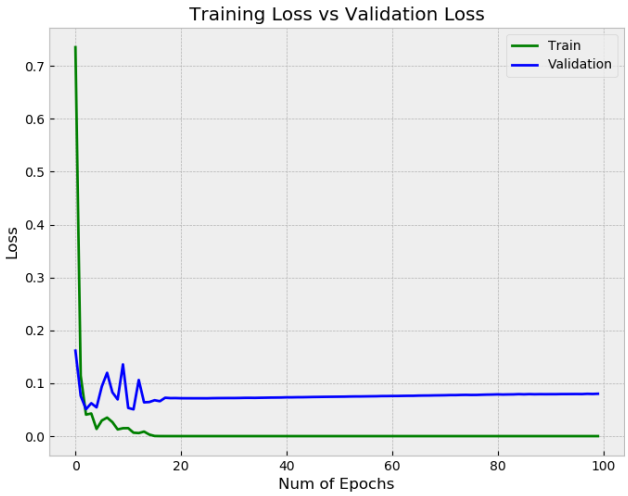

Ilustración 146: Evolución del Loss en la fase de aprendizaje con Alsharif2013 y CJ1

- **Fase de testing**

  La matriz de la Ilustración 147, muestra un resultado muy satisfactorio, ya que ninguna clase despunta en número de errores a la hora de clasificar.

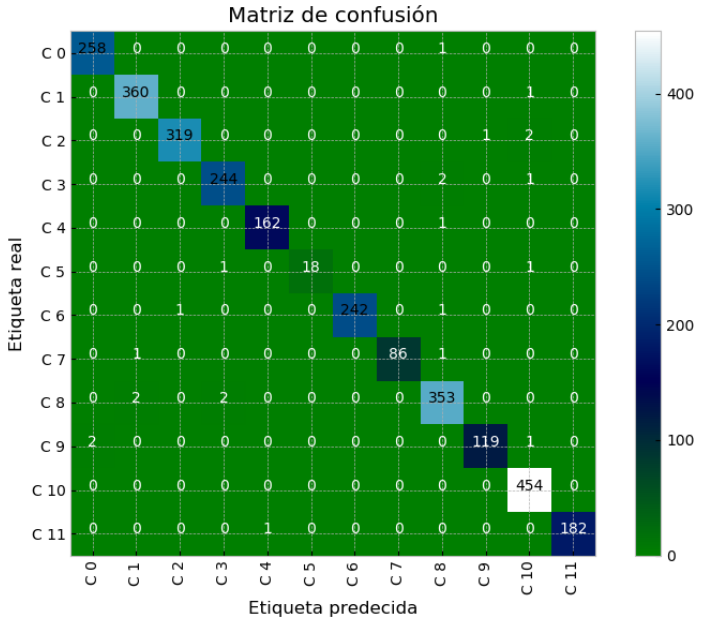

Ilustración 147: Matriz de confusión de la red Alsharif2013 para CJ1

Según los resultados de la Ilustración 148 y la Ilustración 149, la red entra en el intervalo de rendimiento buscado, obteniendo valores cercanos al 99%.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	258.0	2559.0	2.0	1.0	0.9989	0.9961	0.9992	0.9923	0.9996	0.0008	0.0039	0.0077	0.9942
C 1	360.0	2456.0	3.0	1.0	0.9986	0.9972	0.9988	0.9917	0.9996	0.0012	0.0028	0.0083	0.9945
C 2	319.0	2497.0	1.0	3.0	0.9986	0.9907	0.9996	0.9969	0.9988	0.0004	0.0093	0.0031	0.9938
C 3	244.0	2570.0	3.0	3.0	0.9979	0.9879	0.9988	0.9879	0.9988	0.0012	0.0121	0.0121	0.9879
C 4	162.0	2656.0	1.0	1.0	0.9993	0.9939	0.9996	0.9939	0.9996	0.0004	0.0061	0.0061	0.9939
C 5	18.0	2800.0	0.0	2.0	0.9993	0.9	1.0	1.0	0.9993	0.0	0.1	0.0	0.9474
C 6	242.0	2576.0	0.0	2.0	0.9993	0.9918	1.0	1.0	0.9992	0.0	0.0082	0.0	0.9959
C 7	86.0	2732.0	0.0	2.0	0.9993	0.9773	1.0	1.0	0.9993	0.0	0.0227	0.0	0.9885
C 8	353.0	2457.0	6.0	4.0	0.9965	0.9888	0.9976	0.9833	0.9984	0.0024	0.0112	0.0167	0.986
C 9	119.0	2697.0	1.0	3.0	0.9986	0.9754	0.9996	0.9917	0.9989	0.0004	0.0246	0.0083	0.9835
C 10	454.0	2360.0	6.0	0.0	0.9979	1.0	0.9975	0.987	1.0	0.0025	0.0	0.013	0.9934
C 11	182.0	2637.0	0.0	1.0	0.9996	0.9945	1.0	1.0	0.9996	0.0	0.0055	0.0	0.9973

Ilustración 148: Métricas que reflejan el rendimiento del modelo Alsharif2013 para CJ1

ACC	0.9918

Ilustración 149: Accuracy de Alsharif2013 para CJ1

La Curva ROC y el AUC visibles en la Ilustración 150, en esta instancia es del máximo valor.

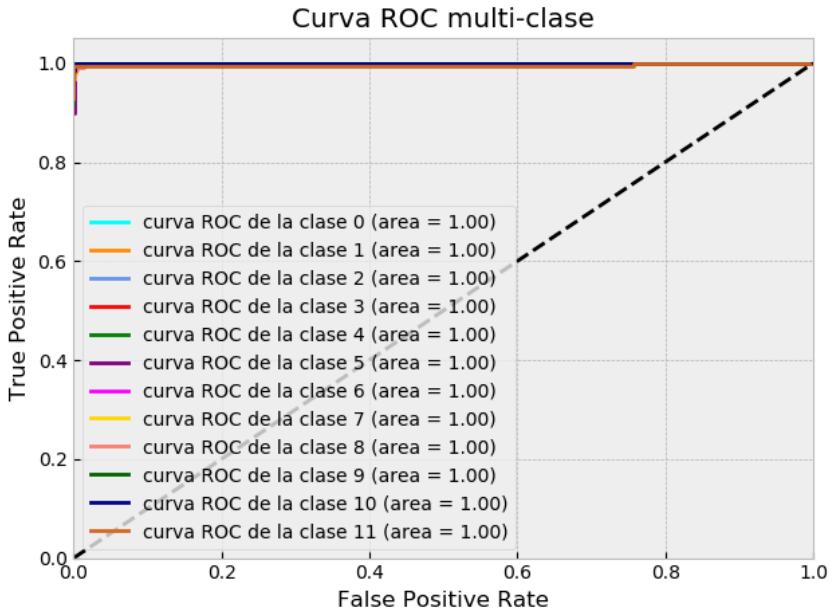

Ilustración 150: Curva ROC y AUC red Alsharif2013 para CJ1

- **Conclusiones**
  Con la arquitectura definida se ha obtenido un ratio de error de **0.81%**, acercándose a los valores obtenidos en la literatura actual, por lo que, es uno de los estudios a valorar para la solución final.

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**
  La evolución del entrenamiento de la red mostrada en la Ilustración 151 y en la Ilustración 152, reflejan un pico de inestabilidad, no obstante, el modelo se estabilizo enseguida y finalmente parece haber obtenido un buen rendimiento.

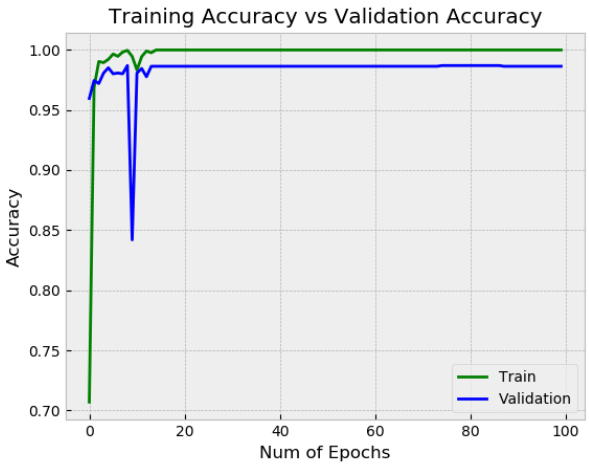

Ilustración 151: Evolución del Accuracy en la fase de aprendizaje con Alsharif2013 y CJ2

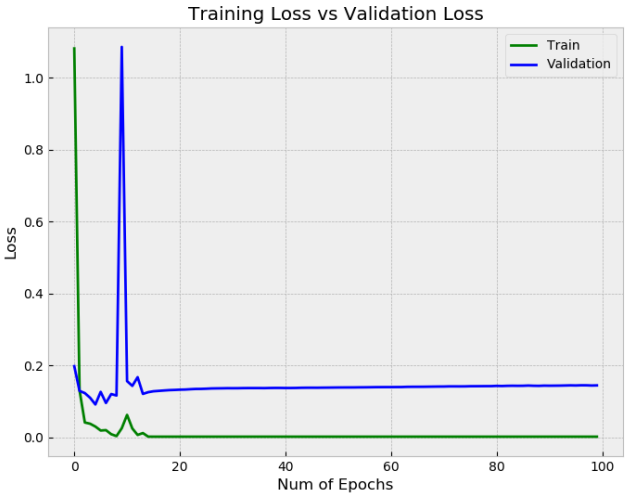

Ilustración 152: Evolución del Loss en la fase de aprendizaje con Alsharif2013 y CJ2

- **Fase de testing**
  La matriz de la Ilustración 153 muestra un buen funcionamiento de la red en la línea del resultado obtenido con CJ1.

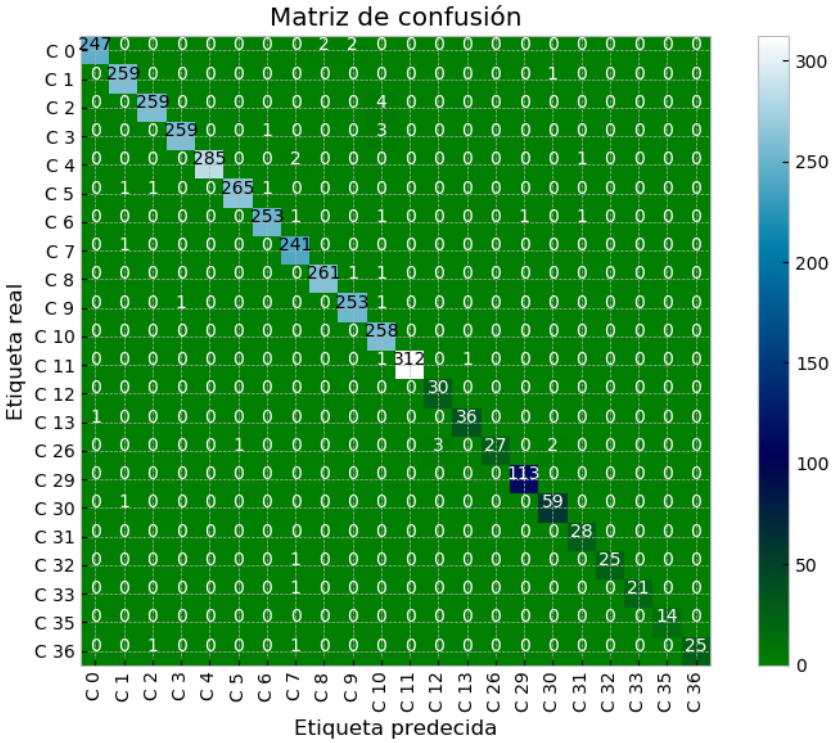

Ilustración 153: Matriz de confusión de la red Alsharif2013 para CJ2

Analizando las métricas obtenidas en la Ilustración 154 y en la Ilustración 155 se avala el buen rendimiento.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	247.0	3320.0	1.0	4.0	0.9986	0.9841	0.9997	0.996	0.9988	0.0003	0.0159	0.004	0.99
C 1	259.0	3309.0	3.0	1.0	0.9989	0.9962	0.9991	0.9885	0.9997	0.0009	0.0038	0.0115	0.9923
C 2	259.0	3307.0	2.0	4.0	0.9983	0.9848	0.9994	0.9923	0.9988	0.0006	0.0152	0.0077	0.9885
C 3	259.0	3308.0	1.0	4.0	0.9986	0.9848	0.9997	0.9962	0.9988	0.0003	0.0152	0.0038	0.9904
C 4	285.0	3284.0	0.0	3.0	0.9992	0.9896	1.0	1.0	0.9991	0.0	0.0104	0.0	0.9948
C 5	265.0	3303.0	1.0	3.0	0.9989	0.9888	0.9997	0.9962	0.9991	0.0003	0.0112	0.0038	0.9925
C 6	253.0	3313.0	2.0	4.0	0.9983	0.9844	0.9994	0.9922	0.9988	0.0006	0.0156	0.0078	0.9883
C 7	241.0	3324.0	6.0	1.0	0.998	0.9959	0.9982	0.9757	0.9997	0.0018	0.0041	0.0243	0.9857
C 8	261.0	3307.0	2.0	2.0	0.9989	0.9924	0.9994	0.9924	0.9994	0.0006	0.0076	0.0076	0.9924
C 9	253.0	3314.0	3.0	2.0	0.9986	0.9922	0.9991	0.9883	0.9994	0.0009	0.0078	0.0117	0.9902
C 10	258.0	3303.0	11.0	0.0	0.9969	1.0	0.9967	0.9591	1.0	0.0033	0.0	0.0409	0.9791
C 11	312.0	3258.0	0.0	2.0	0.9994	0.9936	1.0	1.0	0.9994	0.0	0.0064	0.0	0.9968
C 12	30.0	3539.0	3.0	0.0	0.9992	1.0	0.9992	0.9091	1.0	0.0008	0.0	0.0909	0.9524
C 13	36.0	3534.0	1.0	1.0	0.9994	0.973	0.9997	0.973	0.9997	0.0003	0.027	0.027	0.973
C 26	27.0	3539.0	0.0	6.0	0.9983	0.8182	1.0	1.0	0.9983	0.0	0.1818	0.0	0.9
C 29	113.0	3458.0	1.0	0.0	0.9997	1.0	0.9997	0.9912	1.0	0.0003	0.0	0.0088	0.9956
C 30	59.0	3509.0	3.0	1.0	0.9989	0.9833	0.9991	0.9516	0.9997	0.0009	0.0167	0.0484	0.9672
C 31	28.0	3542.0	2.0	0.0	0.9994	1.0	0.9994	0.9333	1.0	0.0006	0.0	0.0667	0.9655
C 32	25.0	3546.0	0.0	1.0	0.9997	0.9615	1.0	1.0	0.9997	0.0	0.0385	0.0	0.9804
C 33	21.0	3550.0	0.0	1.0	0.9997	0.9545	1.0	1.0	0.9997	0.0	0.0455	0.0	0.9767
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	25.0	3545.0	0.0	2.0	0.9994	0.9259	1.0	1.0	0.9994	0.0	0.0741	0.0	0.9615

Ilustración 154: Métricas que reflejan el rendimiento del modelo Alsharif2013 para CJ2

ACC	0.9882

Ilustración 155: Accuracy de Alsharif2013 para CJ2

Por lo tanto, la Curva ROC y el AUC de la Ilustración 156 son de valor inmejorable.

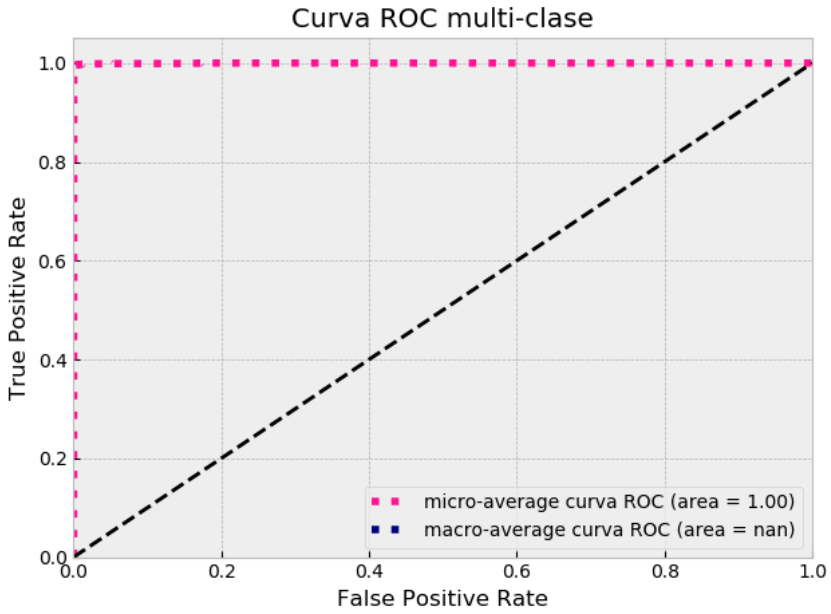

Ilustración 156: Curva ROC y AUC red Alsharif2013 para CJ2

- **Conclusiones**
  La red ha mostrado el mejor desempeño hasta el momento obteniendo un ratio de error de **1.17%**, en este caso no entra dentro del intervalo inferior a 1%, pero es una mejora significativa respecto a las otras redes.

8.1.10 Experimentos con la red Jaderberg2014

El estudio [Jaderberg2014] se centra en el reconocimiento de texto, para ello divide el sistema en subsistemas, centrándose en la tarea de reconocimiento de caracteres al ser el núcleo, aquí se realizará una experimentación utilizando la arquitectura de red de CNN utilizada para dicha tarea. La arquitectura presentada en este estudio es más compleja y utiliza 4 capas de convolución, además, la gran novedad es que usa dropout.

Arquitectura: [Jaderberg2014]

- Numero de capas: **13 capas**
- Capa convolución
  - Número de filtros: **96, 128, 512, 148**
  - Tamaño kernel: **9, 9, 8, 1**
- Capa de pooling

  - Función: **Max**
  - Tamaño: 2
- **Capa de dropout**
  - **Tamaño: 1, 0.5, 0.5, 0.5**
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: **100, 150, 200**

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  El resultado de la fase de aprendizaje de la red se muestra en la Ilustración 157 y en la Ilustración 158, tras observarlas se puede interpretar que la red aún no se ha estabilizado, pudiendo ser que no haya acabado la etapa de aprendizaje. Por ese motivo, se realizó un experimento posterior aumentando el número de epoch con el fin de aclarar si el rendimiento de la red es el mostrado o aún estaba en proceso de aprendizaje.

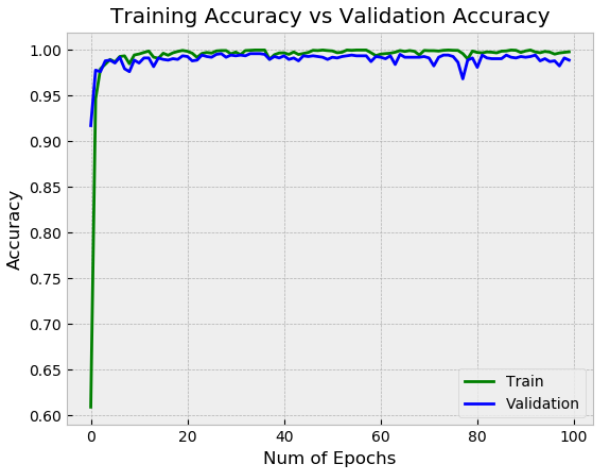

Ilustración 157: Evolución del Accuracy en la fase de aprendizaje con Jaderberg2014 y CJ1 (100 epochs)

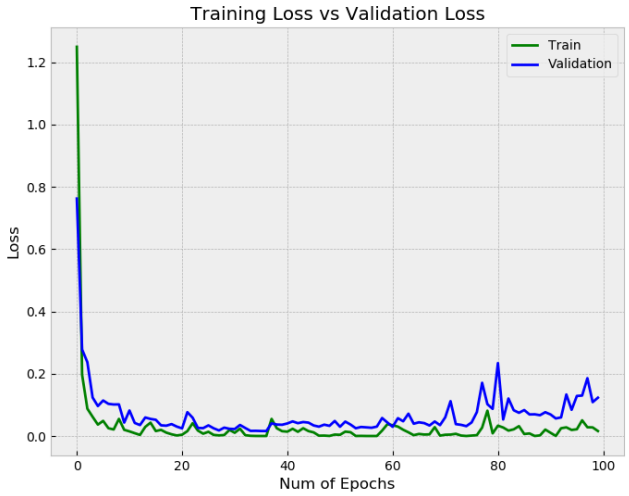

Ilustración 158: Evolución del Loss en la fase de aprendizaje con Jaderberg2014 y CJ1 (100 epochs)

De este modo, el experimento realizado con 150 epochs dio como resultado en la fase aprendizaje el resultado mostrado en la Ilustración 159 y en la Ilustración 160, en esta se

puede observar que la red se ha comportado de manera más constante, por lo que seguiremos analizando el experimento realizado con 150 epochs.

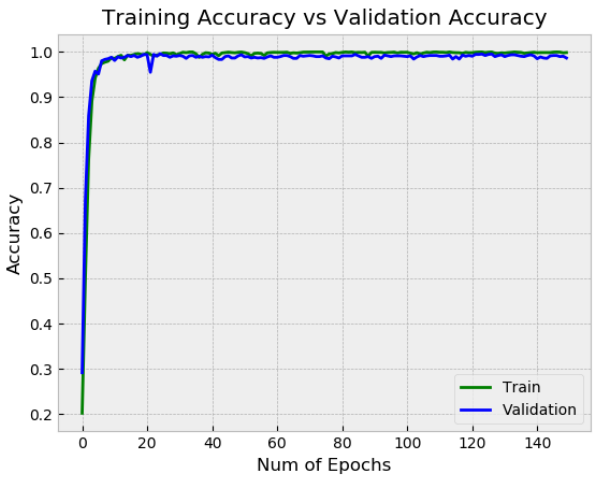

Ilustración 159: Evolución del Accuracy en la fase de aprendizaje con Jaderberg2014 y CJ1 (150 epochs)

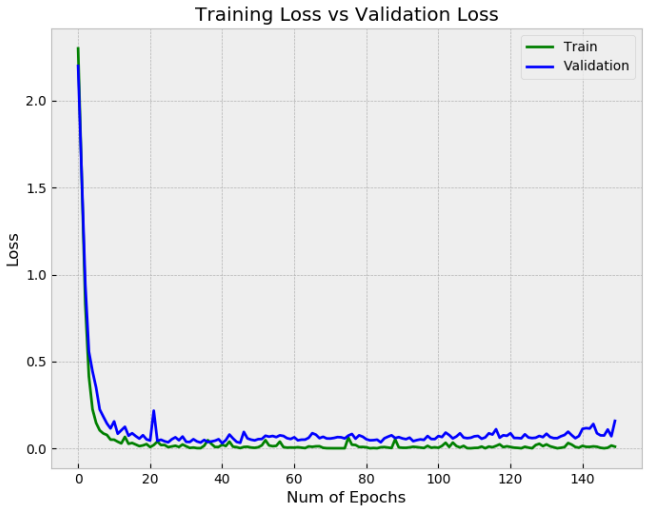

Ilustración 160: Evolución del Loss en la fase de aprendizaje con Jaderberg2014 y CJ1 (150 epochs)

- **Fase de testing**

  Mediante la Ilustración 161, se puede observar que el modelo ha obtenido resultados muy satisfactorios.

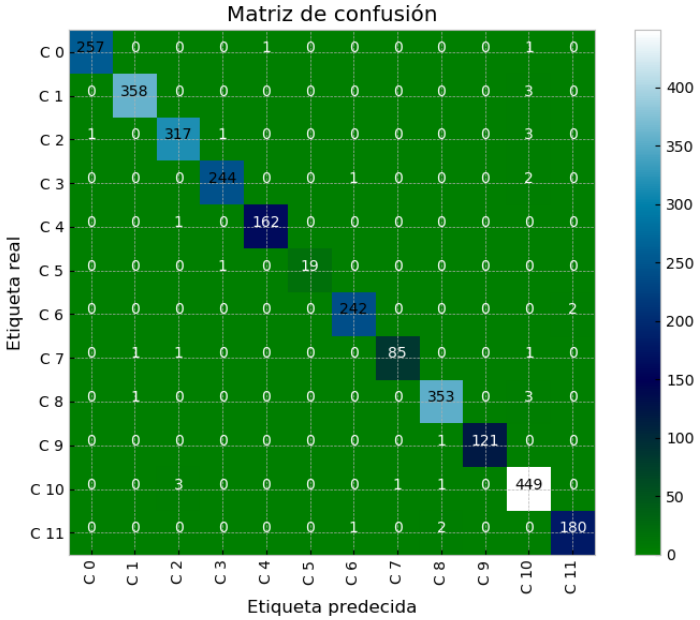

Ilustración 161: Matriz de confusión de la red Jaderberg2014 para CJ1

Los buenos resultados observados en la matriz se pueden corroborar mediante las métricas obtenidas por el modelo en la Ilustración 162 y en la Ilustración 163.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	257.0	2560.0	1.0	2.0	0.9989	0.9923	0.9996	0.9961	0.9992	0.0004	0.0077	0.0039	0.9942
C 1	358.0	2457.0	2.0	3.0	0.9982	0.9917	0.9992	0.9944	0.9988	0.0008	0.0083	0.0056	0.9931
C 2	317.0	2493.0	5.0	5.0	0.9965	0.9845	0.998	0.9845	0.998	0.002	0.0155	0.0155	0.9845
C 3	244.0	2571.0	2.0	3.0	0.9982	0.9879	0.9992	0.9919	0.9988	0.0008	0.0121	0.0081	0.9899
C 4	162.0	2656.0	1.0	1.0	0.9993	0.9939	0.9996	0.9939	0.9996	0.0004	0.0061	0.0061	0.9939
C 5	19.0	2800.0	0.0	1.0	0.9996	0.95	1.0	1.0	0.9996	0.0	0.05	0.0	0.9744
C 6	242.0	2574.0	2.0	2.0	0.9986	0.9918	0.9992	0.9918	0.9992	0.0008	0.0082	0.0082	0.9918
C 7	85.0	2731.0	1.0	3.0	0.9986	0.9659	0.9996	0.9884	0.9989	0.0004	0.0341	0.0116	0.977
C 8	353.0	2459.0	4.0	4.0	0.9972	0.9888	0.9984	0.9888	0.9984	0.0016	0.0112	0.0112	0.9888
C 9	121.0	2698.0	0.0	1.0	0.9996	0.9918	1.0	1.0	0.9996	0.0	0.0082	0.0	0.9959
C 10	449.0	2353.0	13.0	5.0	0.9936	0.989	0.9945	0.9719	0.9979	0.0055	0.011	0.0281	0.9803
C 11	180.0	2635.0	2.0	3.0	0.9982	0.9836	0.9992	0.989	0.9989	0.0008	0.0164	0.011	0.9863

Ilustración 162: Métricas que reflejan el rendimiento del modelo Jaderberg2014 para CJ1

ACC	0.9883

Ilustración 163: Accuracy de Jaderberg2014 para CJ1

De igual manera, la Curva ROC y el AUC tuvieron buenos resultados como se puede observar en la Ilustración 164.

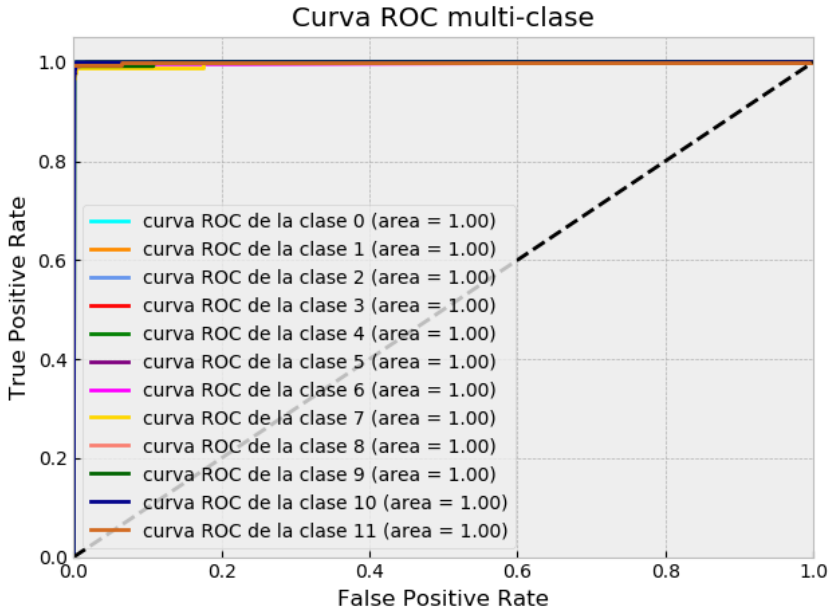

Ilustración 164: Curva ROC y AUC red Jaderberg2014 para CJ1

- **Conclusiones**

  En esta red se ha experimentado con la función de dropout, una elección no muy utilizada para la tarea de reconocimiento de dígitos o caracteres en la literatura

debido a que las redes utilizadas no generan un número de características elevado que haya que simplificar. No obstante, interpretando los experimentos realizados se puede decir que, por una parte, utilizando una red con muchas características como ha sido el caso surge efecto. Pero, por otra parte, se necesitan muchos más recursos computacionales y temporales, ya que fue la red que más tiempo tardo en entrenar, obteniendo un ratio de error del **1.17%** en los dos experimentos realizados, un rendimiento bastante similar al de otras redes que realizan su entrenamiento en mucho menor tiempo.

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  Al igual que sucedió con CJ1, esta red se entrenó inicialmente con 100 epochs al ser considerado generalmente un número adecuado. No obstante, en esta red, como se puede ver en la Ilustración 165 y en la Ilustración 166, la evolución no resulta satisfactoria por lo cual se volvió a realizar el experimento con 200 epoch para probar el funcionamiento de ésta.

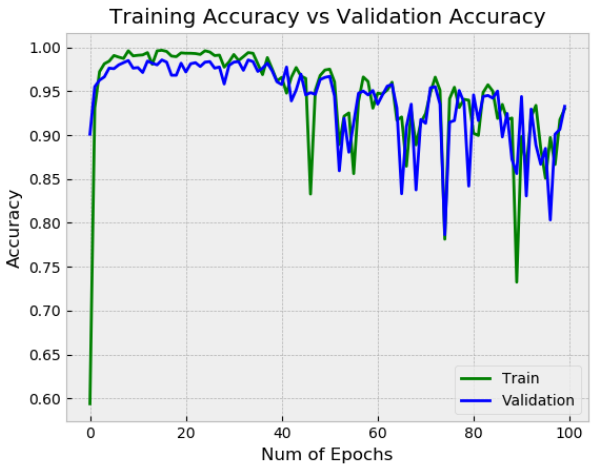

Ilustración 165: Evolución del Accuracy en la fase de aprendizaje con Jaderberg2014 y CJ2 (100 epochs)

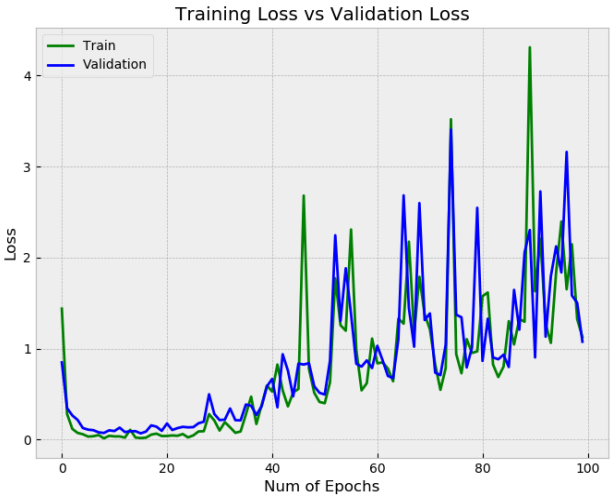

Ilustración 166: Evolución del Loss en la fase de aprendizaje con Jaderberg2014 y CJ2 (100 epochs)

El resultado obtenido en el experimento final de la red se puede observar en la Ilustración 167 y en la Ilustración 168, dejando ver que la opción de incrementar el número de epochs no soluciona el rendimiento del modelo, sino que lo empeora aún más, llegando a unos valores de Accuracy y los nefastos. Por este motivo se analizará el experimento realizado con 100 epochs.

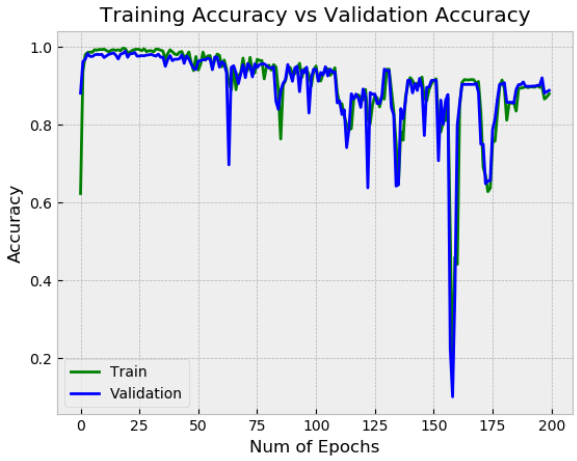

Ilustración 167: Evolución del Accuracy en la fase de aprendizaje con Jaderberg2014 y CJ2 (200 epochs)

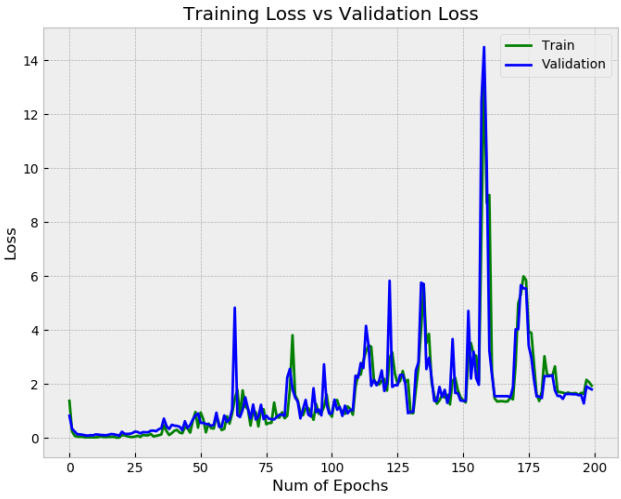

Ilustración 168: Evolución del Loss en la fase de aprendizaje con Jaderberg2014 y CJ2 (200 epochs)

- **Fase de testing**

  El resultado de la matriz de confusión es el de la Ilustración 169. En ella se ve que este modelo retrocede, ya que tiene muchas instancias mal clasificadas

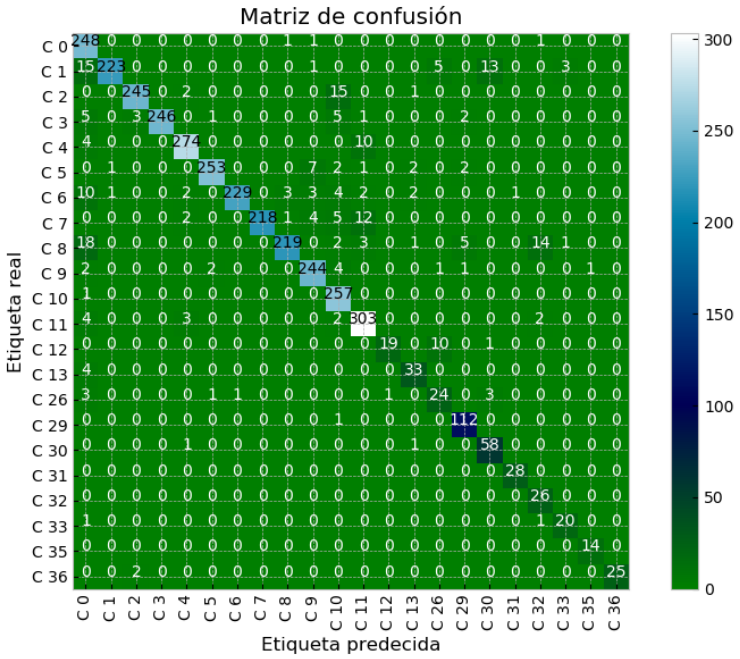

Ilustración 169: Matriz de confusión de la red Jaderberg2014 para CJ2

Como se vió en la matriz, el modelo no obtiene resultados parecidos a los vistos en otras redes hasta ahora, emporando el rendimeinto como se puede ver en la Ilustración 170 y en la Ilustración 171.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	248.0	3254.0	67.0	3.0	0.9804	0.988	0.9798	0.7873	0.9991	0.0202	0.012	0.2127	0.8763
C 1	223.0	3310.0	2.0	37.0	0.9891	0.8577	0.9994	0.9911	0.9889	0.0006	0.1423	0.0089	0.9196
C 2	245.0	3304.0	5.0	18.0	0.9936	0.9316	0.9985	0.98	0.9946	0.0015	0.0684	0.02	0.9552
C 3	246.0	3309.0	0.0	17.0	0.9952	0.9354	1.0	1.0	0.9949	0.0	0.0646	0.0	0.9666
C 4	274.0	3274.0	10.0	14.0	0.9933	0.9514	0.997	0.9648	0.9957	0.003	0.0486	0.0352	0.958
C 5	253.0	3300.0	4.0	15.0	0.9947	0.944	0.9988	0.9844	0.9955	0.0012	0.056	0.0156	0.9638
C 6	229.0	3314.0	1.0	28.0	0.9919	0.8911	0.9997	0.9957	0.9916	0.0003	0.1089	0.0043	0.9405
C 7	218.0	3330.0	0.0	24.0	0.9933	0.9008	1.0	1.0	0.9928	0.0	0.0992	0.0	0.9478
C 8	219.0	3304.0	5.0	44.0	0.9863	0.8327	0.9985	0.9777	0.9869	0.0015	0.1673	0.0223	0.8994
C 9	244.0	3301.0	16.0	11.0	0.9924	0.9569	0.9952	0.9385	0.9967	0.0048	0.0431	0.0615	0.9476
C 10	257.0	3274.0	40.0	1.0	0.9885	0.9961	0.9879	0.8653	0.9997	0.0121	0.0039	0.1347	0.9261
C 11	303.0	3229.0	29.0	11.0	0.9888	0.965	0.9911	0.9127	0.9966	0.0089	0.035	0.0873	0.9381
C 12	19.0	3541.0	1.0	11.0	0.9966	0.6333	0.9997	0.95	0.9969	0.0003	0.3667	0.05	0.76
C 13	33.0	3528.0	7.0	4.0	0.9969	0.8919	0.998	0.825	0.9989	0.002	0.1081	0.175	0.8571
C 26	24.0	3523.0	16.0	9.0	0.993	0.7273	0.9955	0.6	0.9975	0.0045	0.2727	0.4	0.6575
C 29	112.0	3449.0	10.0	1.0	0.9969	0.9912	0.9971	0.918	0.9997	0.0029	0.0088	0.082	0.9532
C 30	58.0	3495.0	17.0	2.0	0.9947	0.9667	0.9952	0.7733	0.9994	0.0048	0.0333	0.2267	0.8593
C 31	28.0	3543.0	1.0	0.0	0.9997	1.0	0.9997	0.9655	1.0	0.0003	0.0	0.0345	0.9825
C 32	26.0	3528.0	18.0	0.0	0.995	1.0	0.9949	0.5909	1.0	0.0051	0.0	0.4091	0.7429
C 33	20.0	3546.0	4.0	2.0	0.9983	0.9091	0.9989	0.8333	0.9994	0.0011	0.0909	0.1667	0.8696
C 35	14.0	3557.0	1.0	0.0	0.9997	1.0	0.9997	0.9333	1.0	0.0003	0.0	0.0667	0.9655
C 36	25.0	3545.0	0.0	2.0	0.9994	0.9259	1.0	1.0	0.9994	0.0	0.0741	0.0	0.9615

Ilustración 170: Métricas que reflejan el rendimiento del modelo Jaderberg2014 para CJ2

ACC	0.9289

Ilustración 171: Accuracy de Jaderberg2014 para CJ2

Por último, el valor de la Curva ROC y el AUC mostrado en la Ilustración 172, también ha sufrido un bajón.

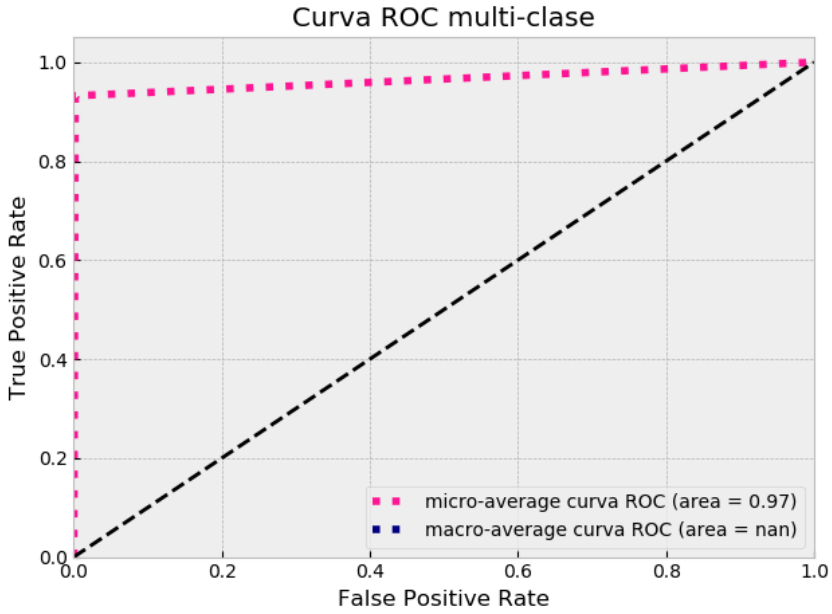

Ilustración 172: Curva ROC y AUC red Jaderberg2014 para CJ2

- **Conclusiones**
  Como ya se concluyó en el experimento de CJ1, además del rendimiento, que para CJ2 ha empeorado bastante obteniendo un ratio de error de **7.11%** para 100 epochs y **11%** para 200 epochs. La red requiere recursos temporales que no se acercan en absoluto a los requeridos por ninguna de las redes probadas hasta ahora.

8.1.11 Conclusiones fase inicial

Tras la realización de la primera fase de experimentación, se puede concluir que hay varias redes en las cuales se puede profundizar y realizar cambios para obtener un mejor rendimiento. A modo de resumen, en la Tabla 15 y la Tabla 16 se presentan los resultados de los experimentos de cada una las redes elegidas en el estudio teórico.

Estas tablas contienen los resultados obtenidos de la última epoch de la fase de entrenamiento, habiendo utilizado el Accuracy, Precisión, Recall y Loss como métricas a optimizar para cada una de las redes, además, son valores representativos de todo el conjunto de clases.

Por otra parte, el ratio de error se ha calculado utilizando el total de instancias que conforman el conjunto test y las instancias mal clasificadas por la red en dicha etapa como se puede observar en (14).

$$Ratio\ error = (1 - (Total - MalClasificadas) / Total) * 100 \qquad (14)$$

Arquitecturas	*Accuracy*	*Precision*	*Recall*	*Loss*	*Tasa Error*
LeNet-5	**0.1276**	**0.0665**	**0.3328**	**5.3650**	**87.23%**
LeNet-5 FunAct	**0.9071**	**0.9104**	**0.9035**	**0.5851**	**9.28%**
LeNet-5 FunActOpt	**0.9883**	**0.9883**	**0.9883**	**0.1636**	**1.16%**
Ranzato2006	**0.9730**	**0.9730**	**0.9730**	**0.3275**	**2.69%**
Simard2003	**0.9578**	**0.9601**	**0.9578**	**0.3616**	**4.07%**
Ciresan2011A	**0.9882**	**0.9882**	**0.9882**	**0.1424**	**1.17%**
Ciresan2011B	**0.9740**	**0.9740**	**0.9726**	**0.2624**	**2.73%**
Chen2015	**0.9886**	**0.9893**	**0.9886**	**0.1083**	**1.13%**
Alsharif2013	**0.9918**	**0.9921**	**0.9918**	**0.0800**	**0.81%**
Jaderberg2014	**0.9882**	**0.9882**	**0.9879**	**0.1104**	**1.17%**

Tabla 15: Resultados de los experimentos con CJ1

Arquitecturas	*Accuracy*	*Precision*	*Recall*	*Loss*	*Tasa Error*
LeNet-5	**0.0736**	**0.0509**	**0.5604**	**11.5200**	**92.63%**
LeNet-5 FunAct	**0.9442**	**0.9482**	**0.9437**	**0.3565**	**5.57%**
LeNet-5FunActOpt	**0.9834**	**0.9834**	**0.9834**	**0.2515**	**1.65%**
Ranzato2006	**0.9582**	**0.9582**	**0.9582**	**0.4425**	**4.17%**
Simard2003	**0.9330**	**0.9341**	**09328**	**0.7239**	**6.69%**
Ciresan2011A	**0.9826**	**0.9831**	**0.9826**	**0.1864**	**1.73%**
Ciresan2011B	**0.9487**	**0.9495**	**0.9482**	**0.4988**	**5.12%**
Chen2015	**0.9846**	**0.9846**	**0.9846**	**0.1610**	**1.53%**
Alsharif2013	**0.9882**	**0.9882**	**0.9882**	**0.1117**	**1.17%**
Jaderberg2014	**0.9288**	**0.9288**	**0.9288**	**1.1419**	**7.11%**

Tabla 16: Resultados de los experimentos con CJ2

Como se puede observar hay tres redes sobre las cuales sería interesante realizar modificaciones en aras de mejorar los resultados ya obtenidos, la denominada LeNet-5 FunActOpt, Chen2015 y Alsharif2013. Por lo tanto, modificando alguna de esas tres redes se obtendrá la solución final al problema planteado.

8.2 Fase final: Experimentos con las redes elegidas

Como se ha planteado, partiendo de las tres redes seleccionas anteriormente se realizará una serie de experimentos, buscando mejorarlas y encontrar el mejor resultado posible.

8.2.1 Experimentos LeNet-5 FunActOpt

Partiendo de la red que se creó modificando la originaria LeNet-5, se decidió seguir experimentando con ella debido al buen rendimiento que se obtenía aun teniendo solo 3 capas de convolución y 2 de submestreo, además de estar utilizando la función average. De este modo, es una red cuya arquitectura se puede mejorar.

Por lo tanto, inicialmente se agregó un nivel con una capa de convolución a la red y se cambió la función de pooling obteniendo una red denominada **LeNet-Plus1** y cuyo modelo quedo así: **6C5-MP5-16C5-MP5-32C5-MP5-120C5-F84-10N.** Esta red obtuvo para **CJ1** un ratio de error de **0.78%** y para **CJ2 2.54%**, unos datos que no son suficientes, por lo tanto se siguió experimentando, creando una red final denominada **LeNet-Plus2**, de la cual se analizarán los resultados a continuación.

Arquitectura: LeNet-Plus2

- Numero de capas: **10 capas**
- Capa convolución

  - Número de filtros: **6, 16, 32, 64**
  - Tamaño kernel: **5**
- Capa de pooling
  - Función: **Max**
  - Tamaño: **5**
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

I. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  Los resultados obtenidos en la fase de entrenamiento y validación para las métricas optimizadas son las mostradas en la Ilustración 173, Ilustración 174, Ilustración 175 y en la Ilustración 176. Estas graficas dejan ver que le modelo alcanza unos valores muy buenos para todas las métricas.

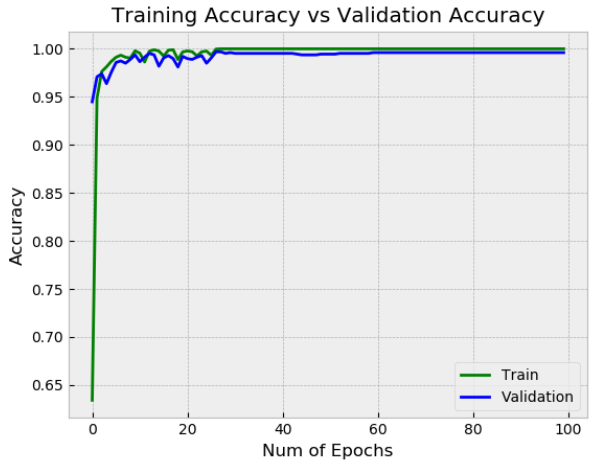

Ilustración 173: Evolución del Accuracy en la fase de aprendizaje con LeNet-Plus2 y CJ1

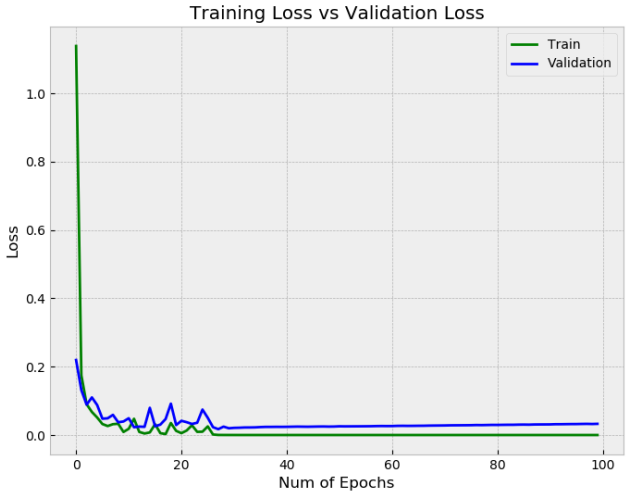

Ilustración 174: Evolución del Loss en la fase de aprendizaje con LeNet-Plus2 y CJ1

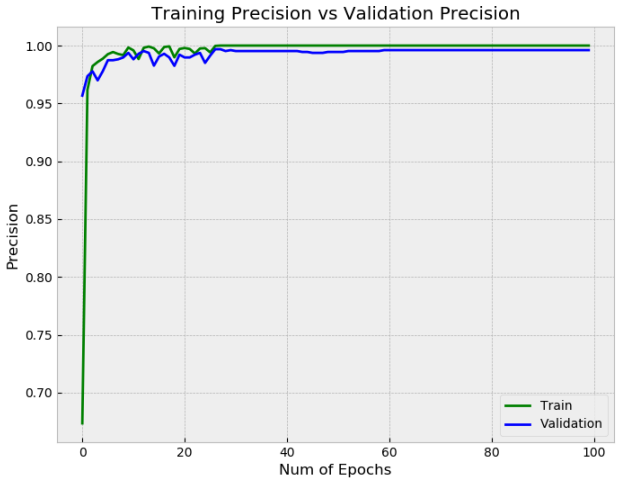

Ilustración 175: Evolución de la Precision en la fase de aprendizaje con LeNet-Plus2 y CJ1

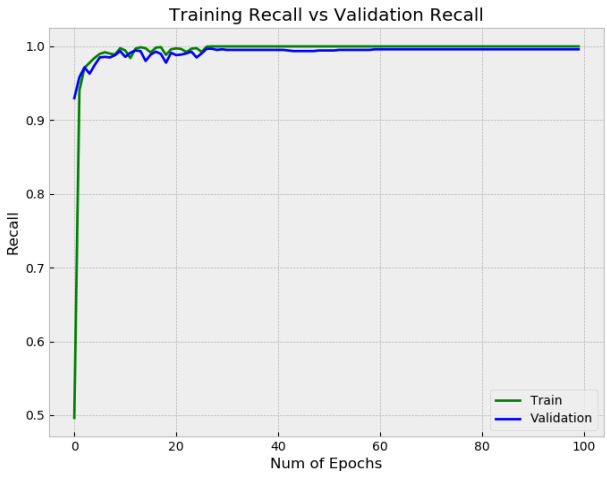

Ilustración 176: Evolución del Recall en la fase de aprendizaje con LeNet-Plus2 y CJ1

- **Fase de testing**
  El testing del modelo arroja la matriz de la Ilustración 177, en ella se puede ver que los resultados del entrenamiento se mantienen en el testing, teniendo muy pocas instancias mal clasificadas.

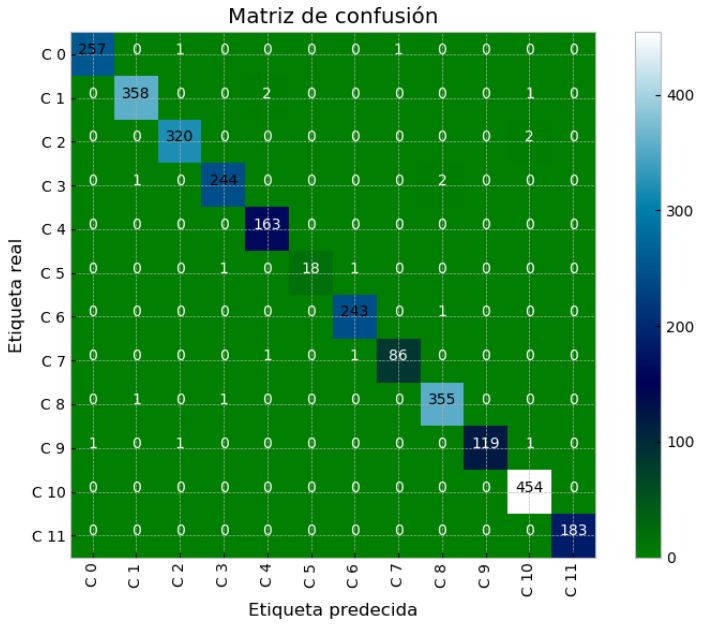

Ilustración 177: Matriz de confusión de la red LeNet-Plus2 para CJ1

De igual modo en las métricas presentadas en la Ilustración 178 y en la Ilustración 179 se alcanza el valor 0.99 para todas las metricas cuyo máximo es 1.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	257.0	2560.0	1.0	2.0	0.9989	0.9923	0.9996	0.9961	0.9992	0.0004	0.0077	0.0039	0.9942
C 1	358.0	2457.0	2.0	3.0	0.9982	0.9917	0.9992	0.9944	0.9988	0.0008	0.0083	0.0056	0.9931
C 2	320.0	2496.0	2.0	2.0	0.9986	0.9938	0.9992	0.9938	0.9992	0.0008	0.0062	0.0062	0.9938
C 3	244.0	2571.0	2.0	3.0	0.9982	0.9879	0.9992	0.9919	0.9988	0.0008	0.0121	0.0081	0.9899
C 4	163.0	2654.0	3.0	0.0	0.9989	1.0	0.9989	0.9819	1.0	0.0011	0.0	0.0181	0.9909
C 5	18.0	2800.0	0.0	2.0	0.9993	0.9	1.0	1.0	0.9993	0.0	0.1	0.0	0.9474
C 6	243.0	2574.0	2.0	1.0	0.9989	0.9959	0.9992	0.9918	0.9996	0.0008	0.0041	0.0082	0.9939
C 7	86.0	2731.0	1.0	2.0	0.9989	0.9773	0.9996	0.9885	0.9993	0.0004	0.0227	0.0115	0.9829
C 8	355.0	2460.0	3.0	2.0	0.9982	0.9944	0.9988	0.9916	0.9992	0.0012	0.0056	0.0084	0.993
C 9	119.0	2698.0	0.0	3.0	0.9989	0.9754	1.0	1.0	0.9989	0.0	0.0246	0.0	0.9876
C 10	454.0	2362.0	4.0	0.0	0.9986	1.0	0.9983	0.9913	1.0	0.0017	0.0	0.0087	0.9956
C 11	183.0	2637.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0

Ilustración 178: Métricas que reflejan el rendimiento del modelo LeNet-Plus2 para CJ1

ACC	0.9929

Ilustración 179: Accuracy de LeNet-Plus2 para CJ1

En el caso de la curva ROC y el AUC del modelo se puede ver en la Ilustración 180 que alcanza el mayor valor posible.

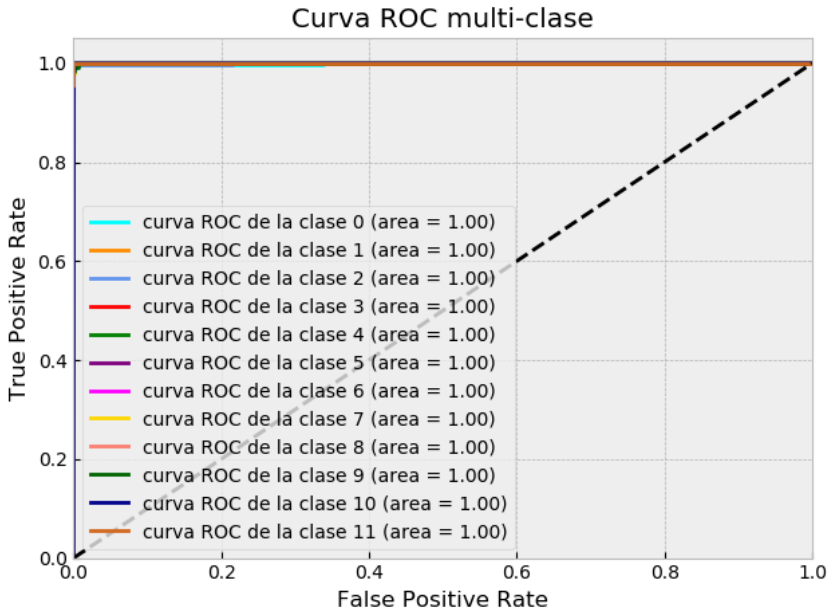

Ilustración 180: Curva ROC y AUC red LeNet-Plus2 para CJ1

Por último, con el objetivo de conocer las características de las imágenes en la que el modelo ha realizado una clasificación errónea, en la Ilustración 181 se puede observar dichas imágenes. Además, están encabezadas por una P y una R, siendo P el valor predicho por la red y R, él valor real que contiene la imágen.
Si se analizan estas imágenes, se puede apreciar que gran parte se trata de imágenes en la que los caracteres están incompletos o poco visibles, confundiendo varias de estas con imágenes en la que solo aparece fondo.

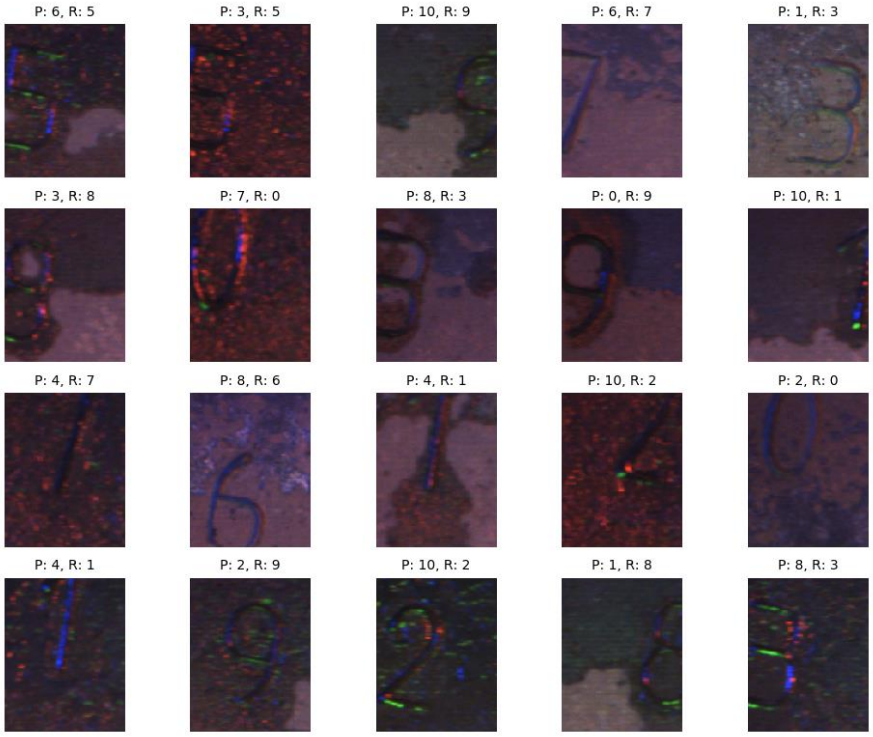

Ilustración 181: Imágenes de CJ1 mal clasificadas por LeNet-Plus2 en la etapa de test

- **Conclusiones**
  La red ha obtenido un ratio de error de **0.70%,** entrando en un intervalo de error menor al 1%, cumpliendo de este modo el objetivo de realizar una clasificación cercana a la obtenida por el ser humano.

II. Conjunto de imágenes CJ2

- **Fase de aprendizaje**
  El resultado de la fase de aprendizaje se puede ver en la Ilustración 182, Ilustración 183, Ilustración 184 y en la Ilustración 185. En ella se puede apreciar que se obtienen buenos resultados.

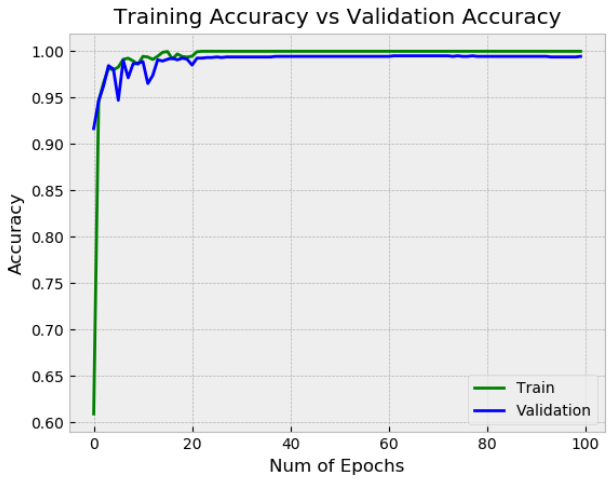

Ilustración 182: Evolución del Accuracy en la fase de aprendizaje con LeNet-Plus2 y CJ2

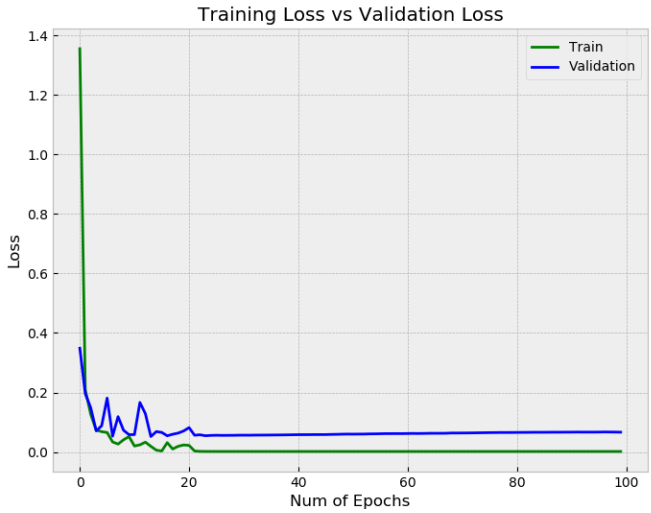

Ilustración 183: Evolución del Loss en la fase de aprendizaje con LeNet-Plus2 y CJ2

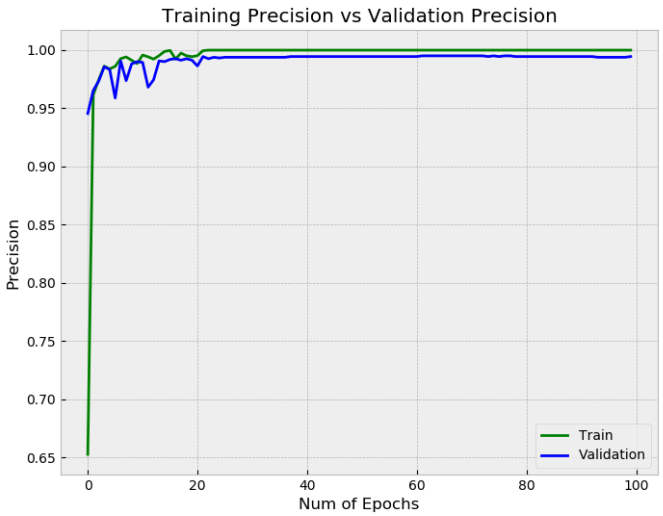

Ilustración 184: Evolución d lal Precision en la fase de aprendizaje con LeNet-Plus2 y CJ2

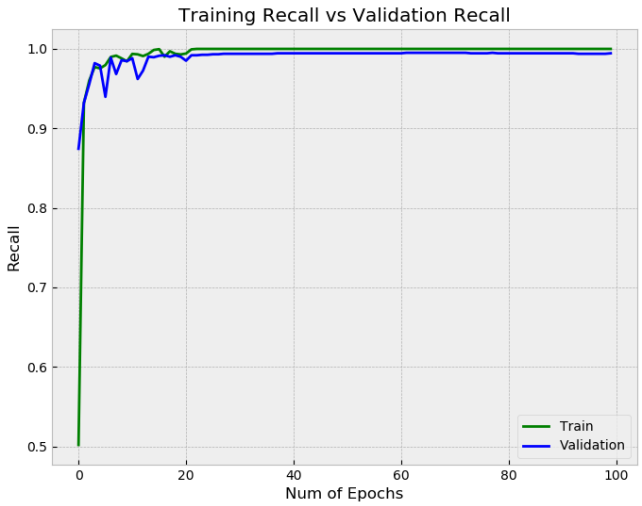

Ilustración 185: Evolución del Recall en la fase de aprendizaje con LeNet-Plus2 y CJ2

- **Fase de testing**

  En cuanto a la matriz de confusión mostrada en la Ilustración 186, se aprecia que la red apenas comete fallos en algunas clases y un número mínimo en las que los comete.

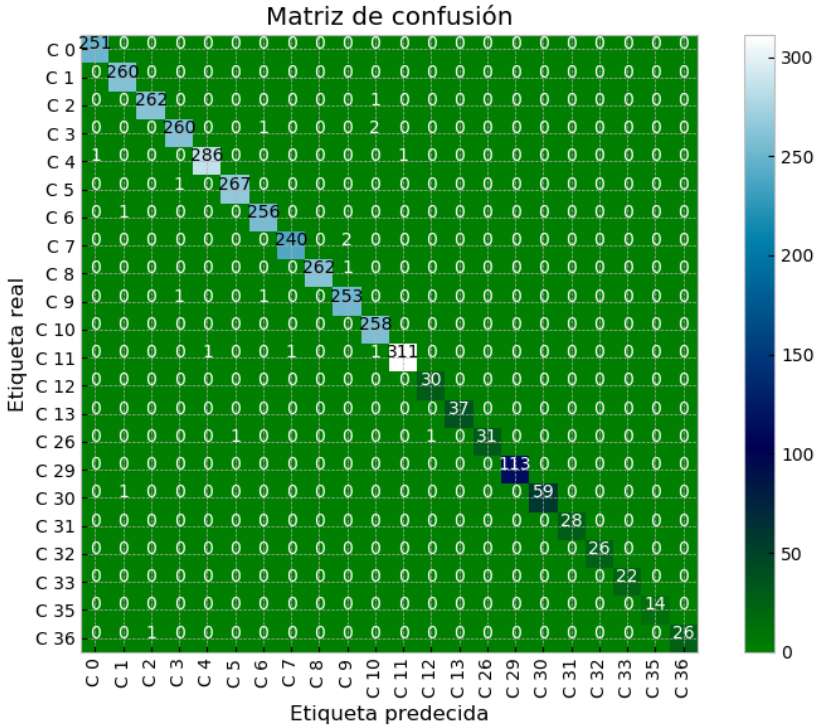

Ilustración 186: Matriz de confusión de la red LeNet-Plus2 para CJ2

Los resultados de la matriz traducidos en métricas se presentan en la Ilustración 187 y en la Ilustración 188. En este caso, también se obtienen valores en los que el margen de error es mínimo.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	251.0	3320.0	1.0	0.0	0.9997	1.0	0.9997	0.996	1.0	0.0003	0.0	0.004	0.998
C 1	260.0	3310.0	2.0	0.0	0.9994	1.0	0.9994	0.9924	1.0	0.0006	0.0	0.0076	0.9962
C 2	262.0	3308.0	1.0	1.0	0.9994	0.9962	0.9997	0.9962	0.9997	0.0003	0.0038	0.0038	0.9962
C 3	260.0	3307.0	2.0	3.0	0.9986	0.9886	0.9994	0.9924	0.9991	0.0006	0.0114	0.0076	0.9905
C 4	286.0	3283.0	1.0	2.0	0.9992	0.9931	0.9997	0.9965	0.9994	0.0003	0.0069	0.0035	0.9948
C 5	267.0	3303.0	1.0	1.0	0.9994	0.9963	0.9997	0.9963	0.9997	0.0003	0.0037	0.0037	0.9963
C 6	256.0	3313.0	2.0	1.0	0.9992	0.9961	0.9994	0.9922	0.9997	0.0006	0.0039	0.0078	0.9942
C 7	240.0	3329.0	1.0	2.0	0.9992	0.9917	0.9997	0.9959	0.9994	0.0003	0.0083	0.0041	0.9938
C 8	262.0	3309.0	0.0	1.0	0.9997	0.9962	1.0	1.0	0.9997	0.0	0.0038	0.0	0.9981
C 9	253.0	3314.0	3.0	2.0	0.9986	0.9922	0.9991	0.9883	0.9994	0.0009	0.0078	0.0117	0.9902
C 10	258.0	3310.0	4.0	0.0	0.9989	1.0	0.9988	0.9847	1.0	0.0012	0.0	0.0153	0.9923
C 11	311.0	3257.0	1.0	3.0	0.9989	0.9904	0.9997	0.9968	0.9991	0.0003	0.0096	0.0032	0.9936
C 12	30.0	3541.0	1.0	0.0	0.9997	1.0	0.9997	0.9677	1.0	0.0003	0.0	0.0323	0.9836
C 13	37.0	3535.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 26	31.0	3539.0	0.0	2.0	0.9994	0.9394	1.0	1.0	0.9994	0.0	0.0606	0.0	0.9688
C 29	113.0	3459.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 30	59.0	3512.0	0.0	1.0	0.9997	0.9833	1.0	1.0	0.9997	0.0	0.0167	0.0	0.9916
C 31	28.0	3544.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 32	26.0	3546.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	26.0	3545.0	0.0	1.0	0.9997	0.963	1.0	1.0	0.9997	0.0	0.037	0.0	0.9811

Ilustración 187: Métricas que reflejan el rendimiento del modelo LeNet-Plus2 para CJ2

ACC	0.9944

Ilustración 188: Accuracy de LeNet-Plus2 para CJ2

De igual modo, la curva ROC y AUC de la Ilustración 189, obtienen su máximo rendimiento.

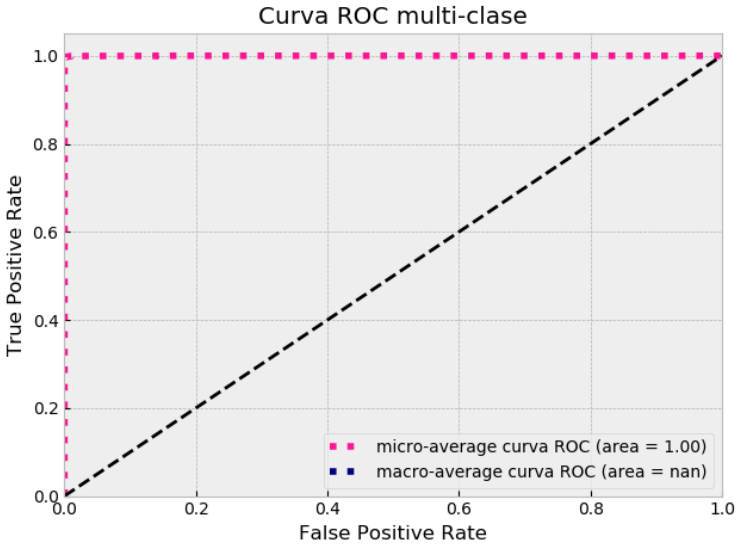

Ilustración 189: Curva ROC y AUC red LeNet-Plus2 para CJ2

Las imágenes cuya clasificación fue errónea se pueden ver en la Ilustración 190. Si se analizan éstas, se puede ver que existe mucha confusión entre letras y dígitos similares y con aquellas imágenes en la que los caracteres están incompletos o borrosos.

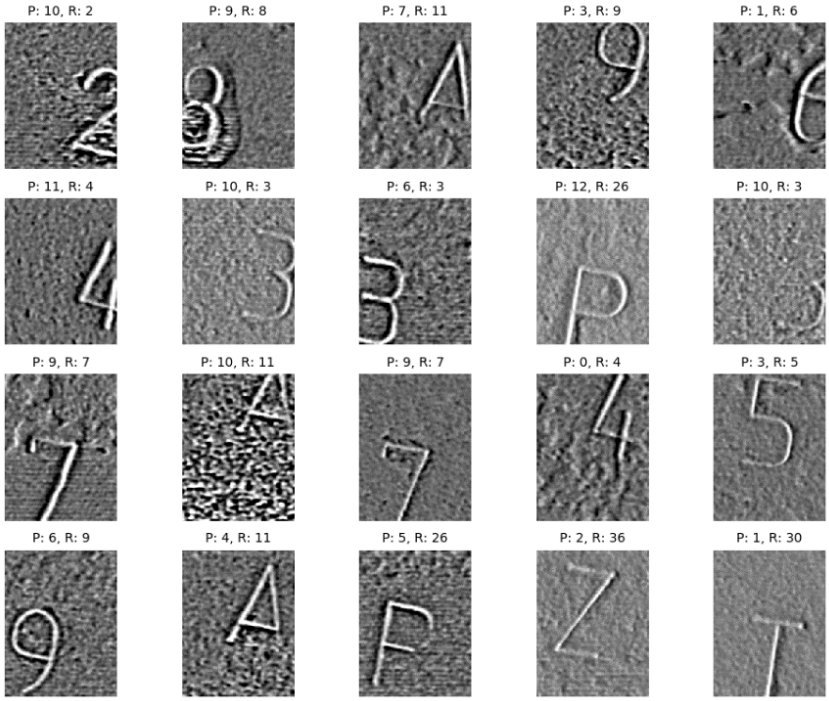

Ilustración 190: Imágenes de CJ2 mal clasificadas por LeNet-Plus2 en la etapa de test

- **Conclusiones**
  En este caso el ratio de error obtenido por la red fue de **0.75%**, obteniendo para este conjunto un rendimiento muy bueno.

8.2.2 Experimentos con la red Chen2015

La experimentación llevada a cabo en esta red consistió en la agregación de un nuevo nivel de convolución, pooling y activación en la primera parte de la red, se jugó con el tamaño del filtro de convolución para obtener uno adecuado, utilizando finalmente uno de 16, de ese modo se obtuvieron mejores resultados.

Arquitectura: Chen-Plus

- Numero de capas: **10 capas**
- Capa convolución
  - Número de filtros: **16, 32, 64, 128**
  - Tamaño kernel: **5, 5, 3, 3**
- Capa de pooling
  - Función: **Max**
  - Tamaño**: 3**
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

III. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  La evolución de la fase de aprendizaje mostrada en la Ilustración 191, la Ilustración 192, la Ilustración 193 y en la Ilustración 194. Deja ver que al igual que en el modelo previo la red creada obtiene un gran rendimiento al alcanzar los intervalos buscados.

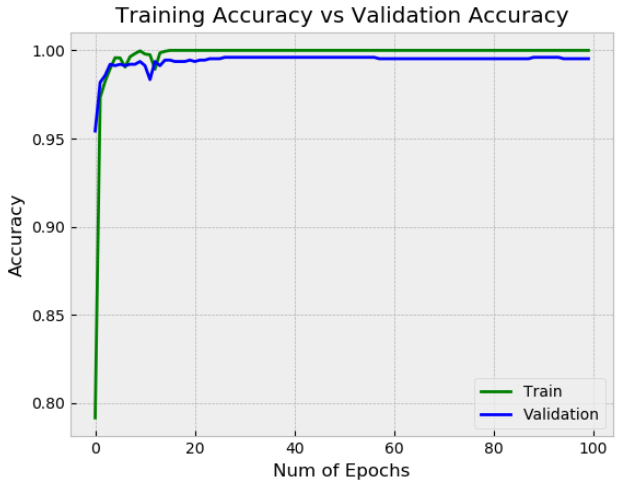

Ilustración 191: Evolución del Accuracy en la fase de aprendizaje con Chen-Plus y CJ1

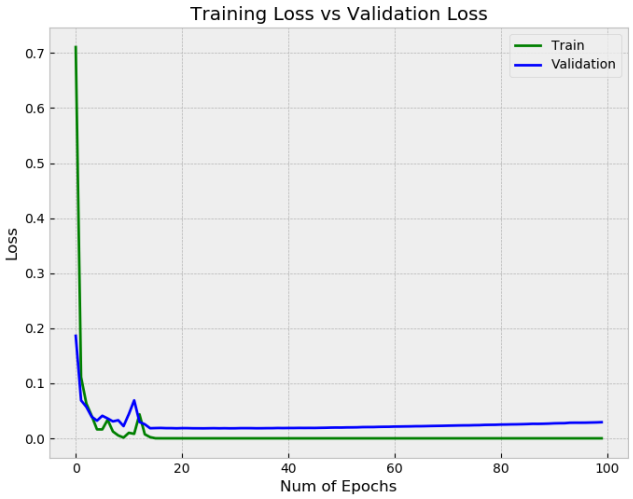

Ilustración 192: Evolución del Loss en la fase de aprendizaje con Chen-Plus y CJ1

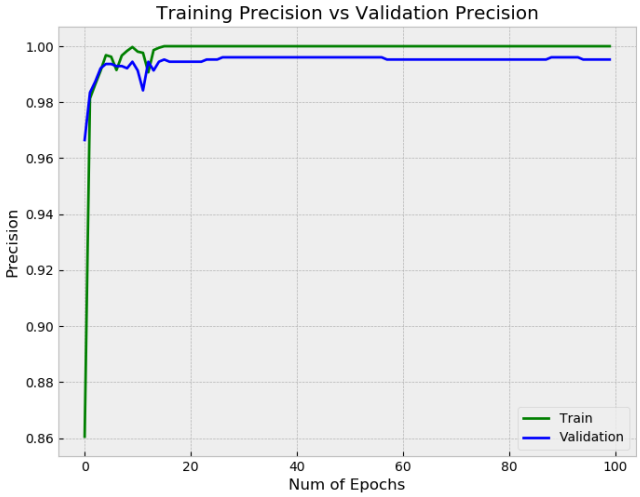

Ilustración 193: Evolución de la Precision en la fase de aprendizaje con Chen-Plus y CJ1

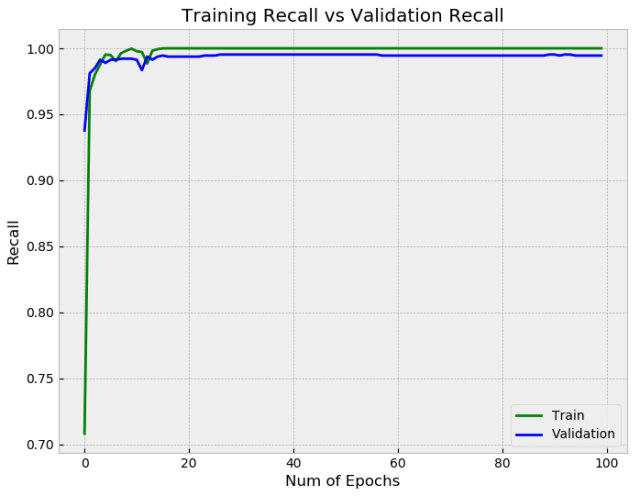

Ilustración 194: Evolución del Recall en la fase de aprendizaje con Chen-Plus y CJ1

- **Fase de testing**
  La matriz de la Ilustración 195, continúa indicando que se trata de una red cuyo rendimiento entra dentro del rendimiento existente en el estado de la técnica.

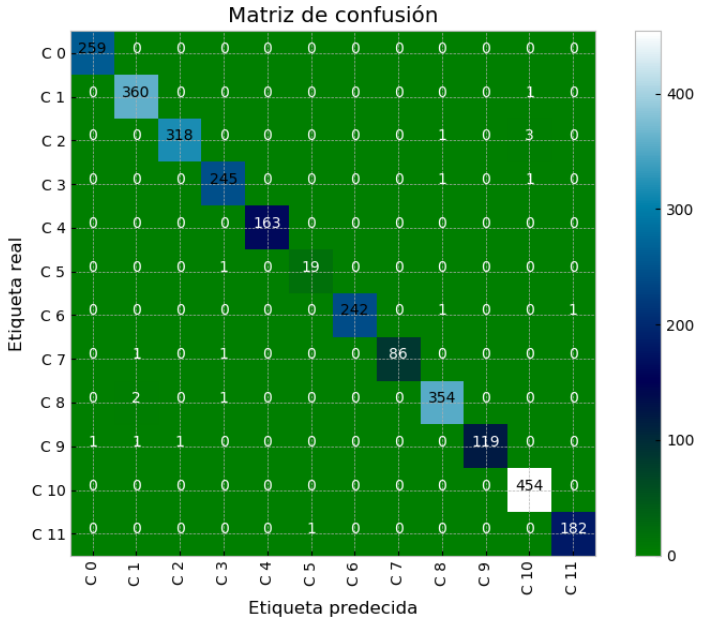

Ilustración 195: Matriz de confusión de la red Chen-Plus para CJ1

De este modo, las métricas de la red mostradas en la Ilustración 196 y en la Ilustración 197, denotan un rendimiento muy similar al conseguido con la red previa.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	259.0	2560.0	1.0	0.0	0.9996	1.0	0.9996	0.9962	1.0	0.0004	0.0	0.0038	0.9981
C 1	360.0	2455.0	4.0	1.0	0.9982	0.9972	0.9984	0.989	0.9996	0.0016	0.0028	0.011	0.9931
C 2	318.0	2497.0	1.0	4.0	0.9982	0.9876	0.9996	0.9969	0.9984	0.0004	0.0124	0.0031	0.9922
C 3	245.0	2570.0	3.0	2.0	0.9982	0.9919	0.9988	0.9879	0.9992	0.0012	0.0081	0.0121	0.9899
C 4	163.0	2657.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 5	19.0	2799.0	1.0	1.0	0.9993	0.95	0.9996	0.95	0.9996	0.0004	0.05	0.05	0.95
C 6	242.0	2576.0	0.0	2.0	0.9993	0.9918	1.0	1.0	0.9992	0.0	0.0082	0.0	0.9959
C 7	86.0	2732.0	0.0	2.0	0.9993	0.9773	1.0	1.0	0.9993	0.0	0.0227	0.0	0.9885
C 8	354.0	2460.0	3.0	3.0	0.9979	0.9916	0.9988	0.9916	0.9988	0.0012	0.0084	0.0084	0.9916
C 9	119.0	2698.0	0.0	3.0	0.9989	0.9754	1.0	1.0	0.9989	0.0	0.0246	0.0	0.9876
C 10	454.0	2361.0	5.0	0.0	0.9982	1.0	0.9979	0.9891	1.0	0.0021	0.0	0.0109	0.9945
C 11	182.0	2636.0	1.0	1.0	0.9993	0.9945	0.9996	0.9945	0.9996	0.0004	0.0055	0.0055	0.9945

Ilustración 196: Métricas que reflejan el rendimiento del modelo Chen-Plus para CJ1

ACC	0.9933

Ilustración 197: Accuracy de Chen-Plus para CJ1

En cuanto a la curva ROC y el AUC de la Ilustración 198, en efecto también alcanza el mayor valor posible.

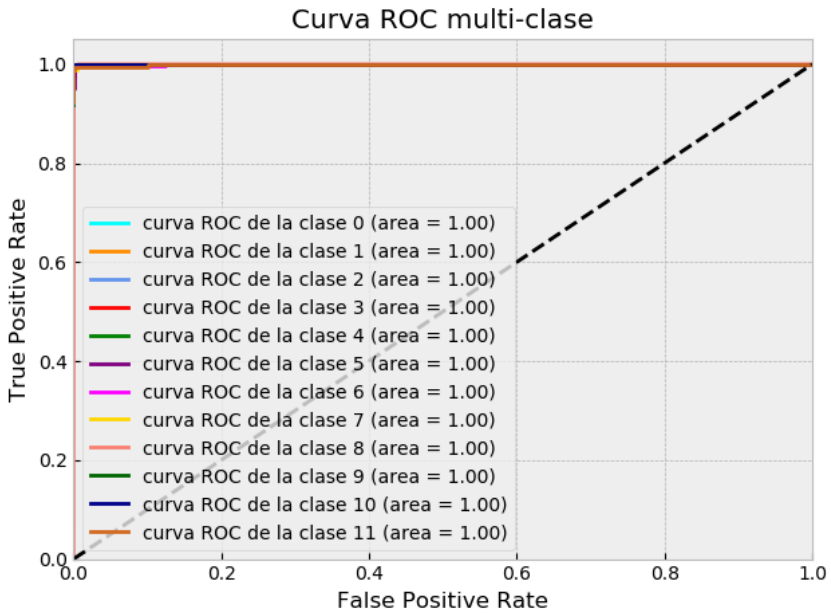

Ilustración 198: Curva ROC y AUC red Chen-Plus para CJ1

Por tanto, lo último que se debe analizar es la Ilustración 199. En ella se puede ver que casi la totalidad de las imágenes mal clasificadas no se encuentran en condiciones normales, es decir, o los caracteres estan incompletos o interfiere alguna característica de calidad como puede ser la iluminación.

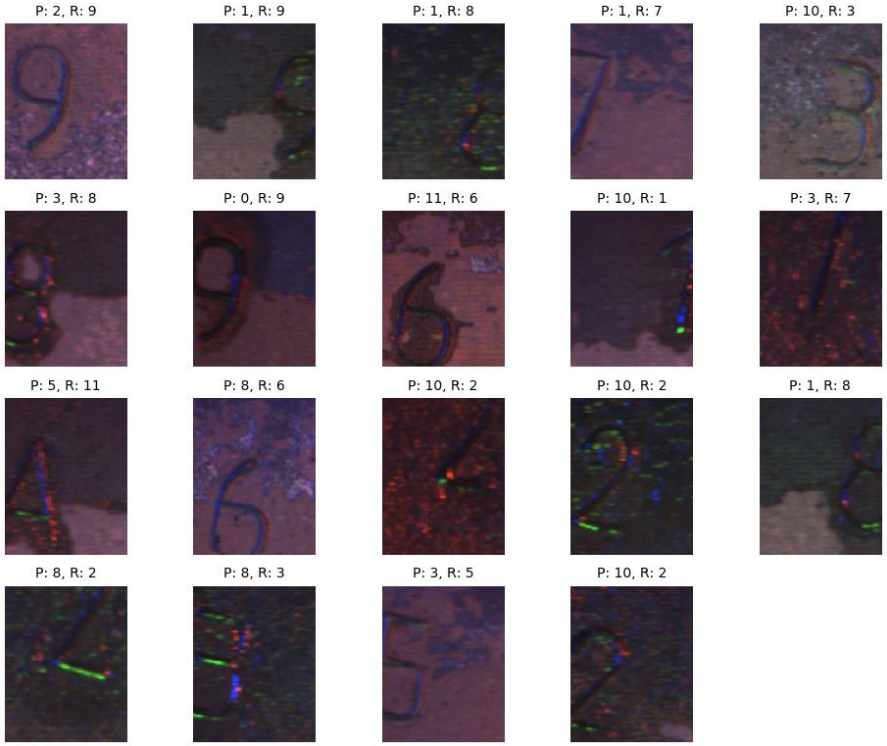

Ilustración 199: Imágenes de CJ1 mal clasificadas por Chen-Plus en la etapa de test

- **Conclusiones**

  En este caso, la red ha obtenido un ratio de error del **0.67%,** superando en muy pequeño margen a la red probada en el experimento anterior.

IV. Conjunto de imágenes CJ2

- **Fase de aprendizaje**

  El resultado de la fase de aprendizaje de la red se puede ver en la Ilustración 200, en la Ilustración 201, en la Ilustración 202 y en la Ilustración 203. La evolución refleja una buena tónica de rendimiento, aunque en la fase final se puede ver como la función de pérdida sufre una pequeña subida que puede tener repercusión en el rendimiento final.

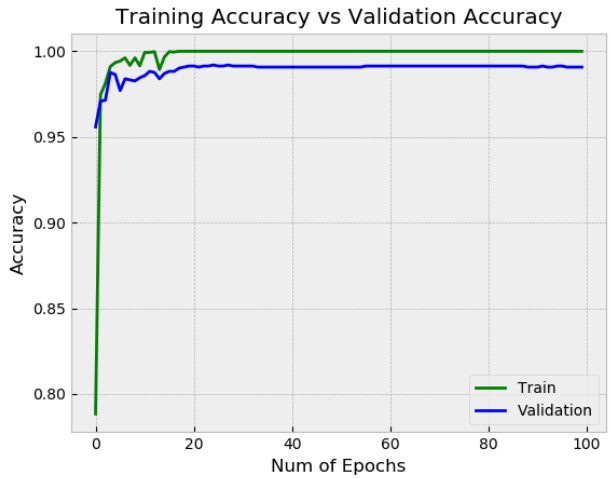

Ilustración 200: Evolución del Accuracy en la fase de aprendizaje con Chen-Plus y CJ2

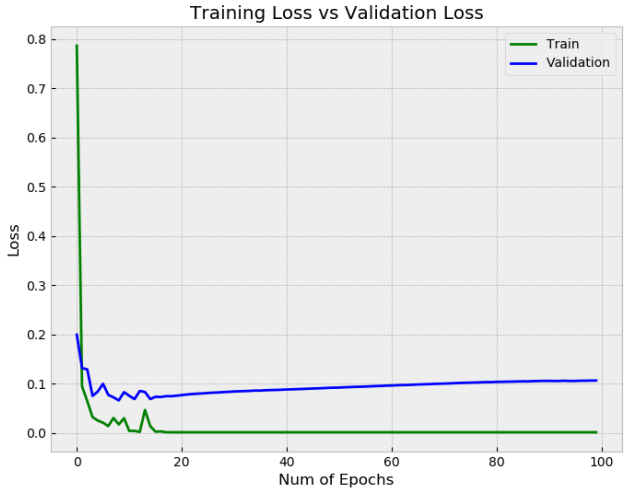

Ilustración 201: Evolución del Loss en la fase de aprendizaje con Chen-Plus y CJ2

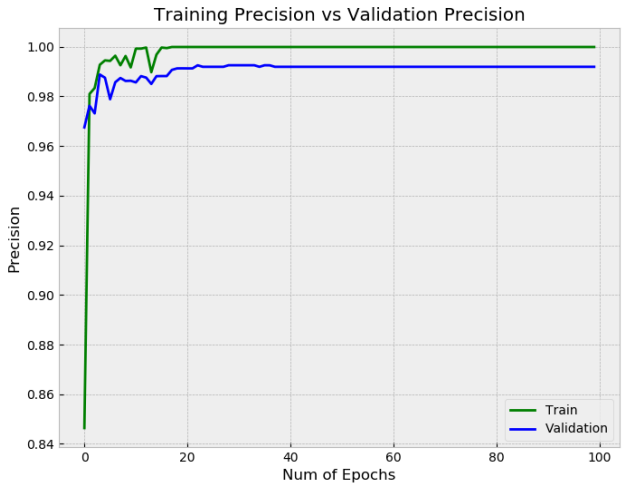

Ilustración 202: Evolución de la Precision en la fase de aprendizaje con Chen-Plus y CJ2

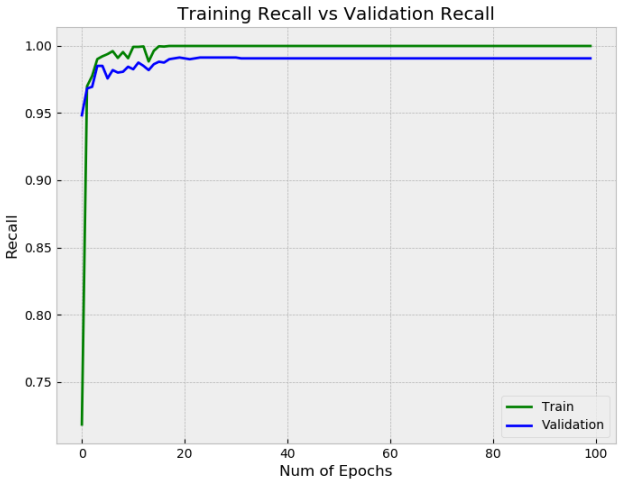

Ilustración 203: Evolución del Recall en la fase de aprendizaje con Chen-Plus y CJ2

- **Fase de testing**
  Analizando la matriz de la Ilustración 204, se puede ver que para algunas clases hubo numerosas instancias mal clasificadas.

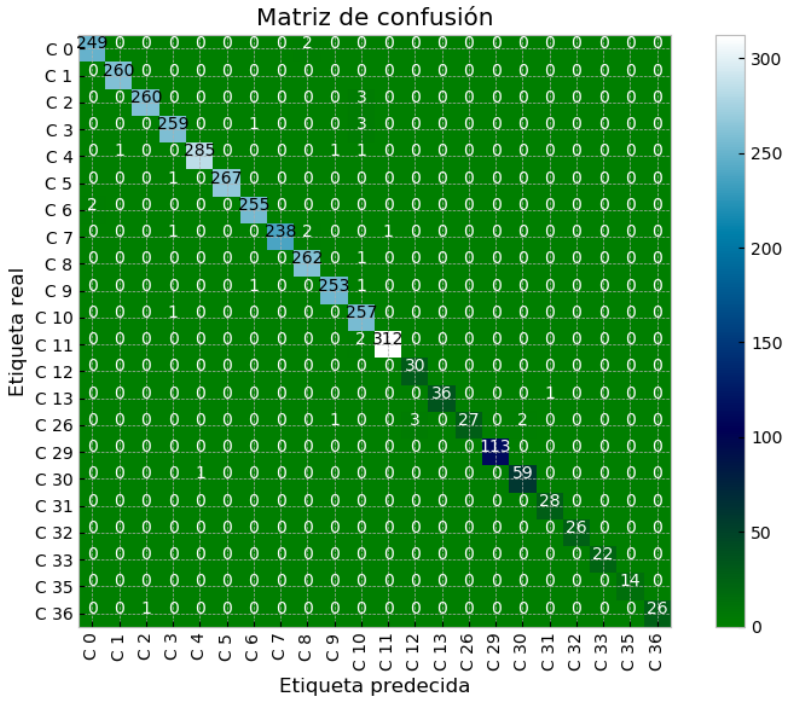

Ilustración 204: Matriz de confusión de la red Chen-Plus para CJ2

Siendo así, las métricas de la Ilustración 205 y la Ilustración 206 reflejan un pequeño bajón de rendimiento respecto a la red previa, encontrande en el filo de 0.99.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	249.0	3319.0	2.0	2.0	0.9989	0.992	0.9994	0.992	0.9994	0.0006	0.008	0.008	0.992
C 1	260.0	3311.0	1.0	0.0	0.9997	1.0	0.9997	0.9962	1.0	0.0003	0.0	0.0038	0.9981
C 2	260.0	3308.0	1.0	3.0	0.9989	0.9886	0.9997	0.9962	0.9991	0.0003	0.0114	0.0038	0.9924
C 3	259.0	3306.0	3.0	4.0	0.998	0.9848	0.9991	0.9885	0.9988	0.0009	0.0152	0.0115	0.9867
C 4	285.0	3283.0	1.0	3.0	0.9989	0.9896	0.9997	0.9965	0.9991	0.0003	0.0104	0.0035	0.993
C 5	267.0	3304.0	0.0	1.0	0.9997	0.9963	1.0	1.0	0.9997	0.0	0.0037	0.0	0.9981
C 6	255.0	3313.0	2.0	2.0	0.9989	0.9922	0.9994	0.9922	0.9994	0.0006	0.0078	0.0078	0.9922
C 7	238.0	3330.0	0.0	4.0	0.9989	0.9835	1.0	1.0	0.9988	0.0	0.0165	0.0	0.9917
C 8	262.0	3305.0	4.0	1.0	0.9986	0.9962	0.9988	0.985	0.9997	0.0012	0.0038	0.015	0.9905
C 9	253.0	3315.0	2.0	2.0	0.9989	0.9922	0.9994	0.9922	0.9994	0.0006	0.0078	0.0078	0.9922
C 10	257.0	3303.0	11.0	1.0	0.9966	0.9961	0.9967	0.959	0.9997	0.0033	0.0039	0.041	0.9772
C 11	312.0	3257.0	1.0	2.0	0.9992	0.9936	0.9997	0.9968	0.9994	0.0003	0.0064	0.0032	0.9952
C 12	30.0	3539.0	3.0	0.0	0.9992	1.0	0.9992	0.9091	1.0	0.0008	0.0	0.0909	0.9524
C 13	36.0	3535.0	0.0	1.0	0.9997	0.973	1.0	1.0	0.9997	0.0	0.027	0.0	0.9863
C 26	27.0	3539.0	0.0	6.0	0.9983	0.8182	1.0	1.0	0.9983	0.0	0.1818	0.0	0.9
C 29	113.0	3459.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 30	59.0	3510.0	2.0	1.0	0.9992	0.9833	0.9994	0.9672	0.9997	0.0006	0.0167	0.0328	0.9752
C 31	28.0	3543.0	1.0	0.0	0.9997	1.0	0.9997	0.9655	1.0	0.0003	0.0	0.0345	0.9825
C 32	26.0	3546.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	26.0	3545.0	0.0	1.0	0.9997	0.963	1.0	1.0	0.9997	0.0	0.037	0.0	0.9811

Ilustración 205: Métricas que reflejan el rendimiento del modelo Chen-Plus para CJ2

ACC	0.9905

Ilustración 206: Accuracy de Chen-Plus para CJ2

En la curva ROC y el AUC de la Ilustración 207 dicho bajón de rendimiento no se notó.

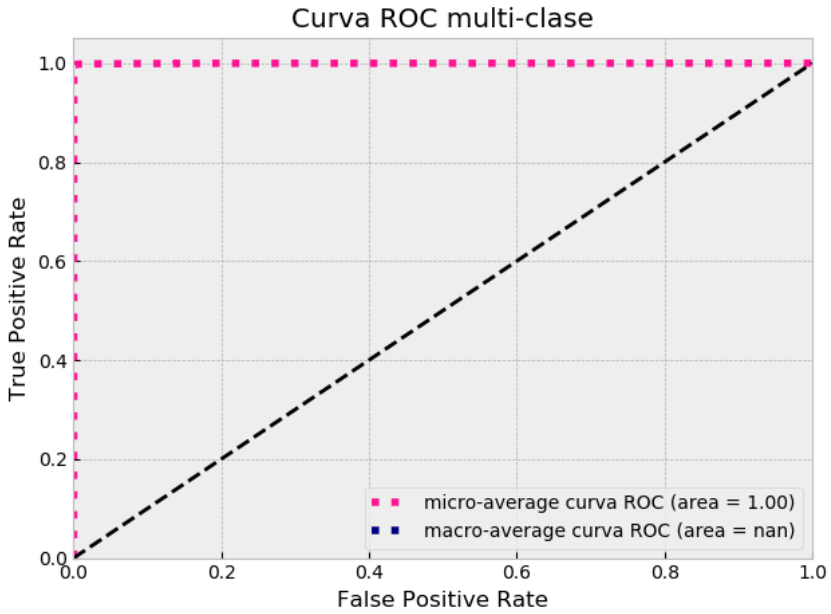

Ilustración 207: Curva ROC y AUC red Chen-Plus para CJ2

No obstante, en la Ilustración 208, se puede ver que ha habido un incremento en el número de imágenes mal clasificadas siendo dicho aumento causado principalmente por imágenes cuyo carácter no está completo, siendo clasificado como una imagen que contiene únicamente fondo. Esto puede ser debido al preprocesamiento que presentan, ya que las imágenes tienen unos rasgos más marcados y al no encontrarse alguna característica se produjo la confusión.

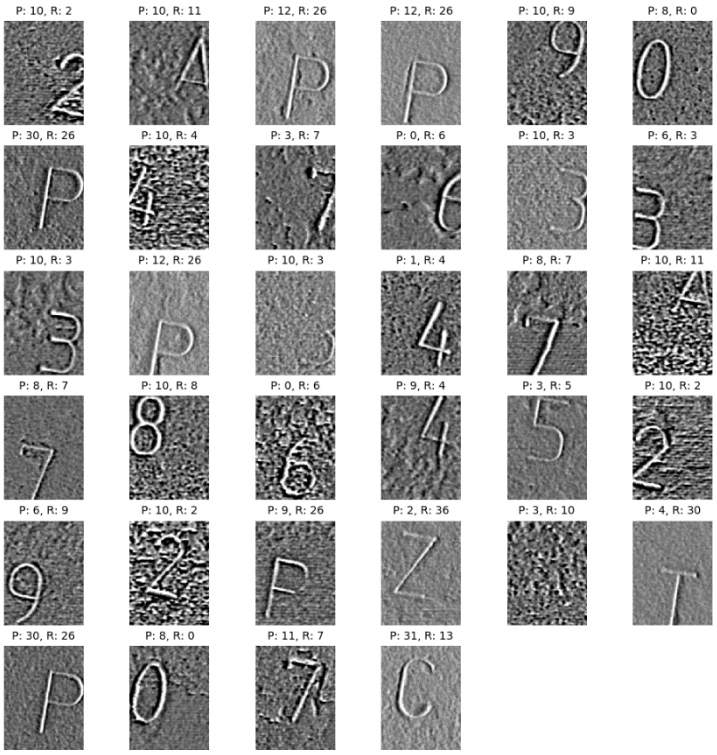

Ilustración 208: Imágenes de CJ2 mal clasificadas por Chen-Plus

- **Conclusiones**
  Con esta red se ha obtenido un ratio de error de **0.95**%, significando un número de errores mayor al conseguido con la red previa, y por lo tanto un rendimiento menor.

8.2.3 Experimentos con la red Alsharif2013

Al igual que pasó con la red de Chen2015, en esta red se experimentó con la adición de un nivel inicial en la arquitectura que se tenía, e igualmente se experimentó con el valor de los filtros de la capa de convolución hasta obtener la que mejores resultados daba.

Arquitectura: Alsharif-Plus

- Numero de capas: **10 capas**
- Capa convolución
  - Número de filtros: **18, 48, 128, 128**
  - Tamaño kernel: **4**, **4, 8, 5**
- Capa de pooling
  - Función: **Max**
  - Tamaño: **4**
- Capa de activación
  - Función: ReLu
- Capa de salida:
  - Función: Softmax

Hiperparámetros

- Función de optimización: Adam
- Learning rate: 1.0x10^-3
- Epochs: 100

Resultados

V. Conjunto de imágenes CJ1

- **Fase de aprendizaje**

  En la evolución del modelo presentado en la Ilustración 209, en la Ilustración 210, en la Ilustración 211 y en la Ilustración 212, se deja ver que la red obtiene un rendimiento que en principio parece superior al obtenido en las redes hasta ahora probadas.

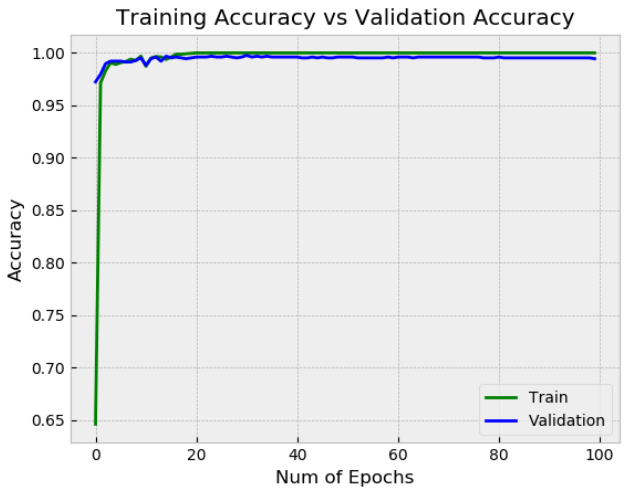

Ilustración 209: Evolución del Accuracy en la fase de aprendizaje con Alsharif-Plus y CJ1

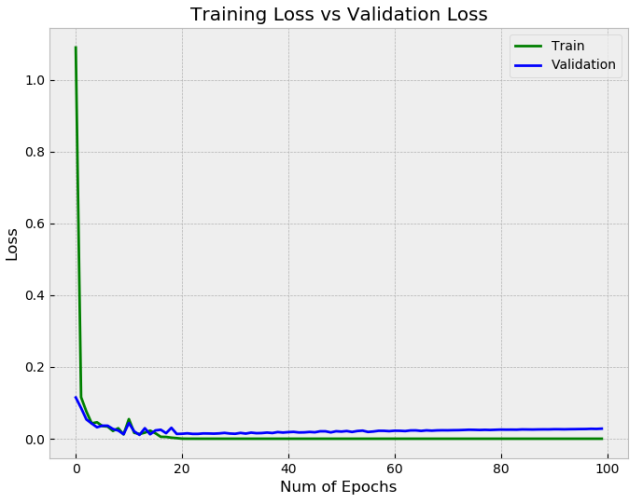

Ilustración 210: Evolución del Loss en la fase de aprendizaje con Alsharif-Plus y CJ1

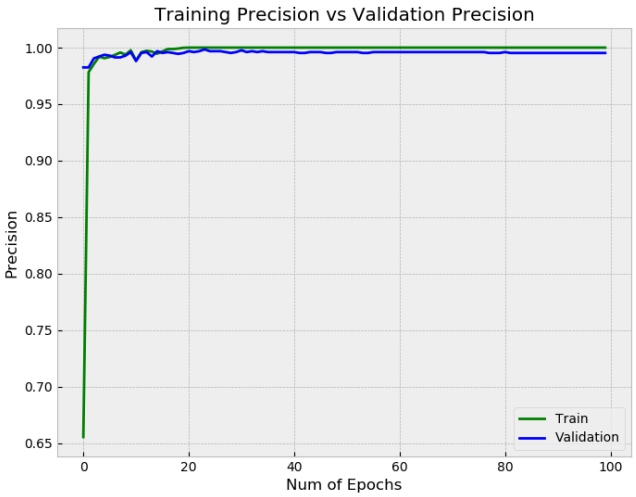

Ilustración 211: Evolución de la Precision en la fase de aprendizaje con Alsharif-Plus y CJ1

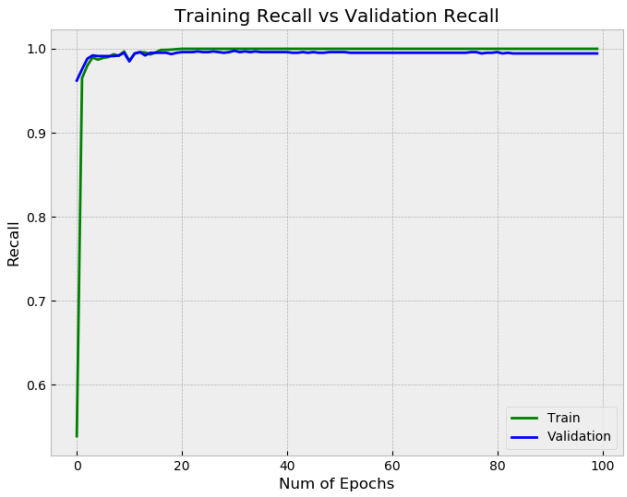

Ilustración 212: Evolución del Recall en la fase de aprendizaje con Alsharif-Plus y CJ1

- **Fase de testing**

  La matriz de confusión del modelo es la mostrada en la Ilustración 213**.**

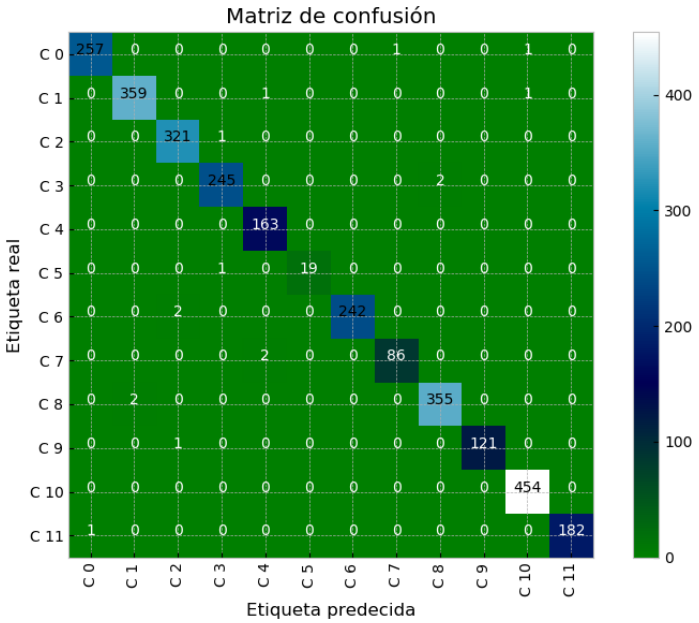

Ilustración 213: Matriz de confusión de la red Alsharif-Plus para CJ1

Y las métricas obtenidas a partir de la matriz se encuentran en la Ilustración 214 y en la Ilustración 215. Si éstas son analizadas se puede ver la pequeña mejoría observada en la fase de aprendizaje.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	257.0	2560.0	1.0	2.0	0.9989	0.9923	0.9996	0.9961	0.9992	0.0004	0.0077	0.0039	0.9942
C 1	359.0	2457.0	2.0	2.0	0.9986	0.9945	0.9992	0.9945	0.9992	0.0008	0.0055	0.0055	0.9945
C 2	321.0	2495.0	3.0	1.0	0.9986	0.9969	0.9988	0.9907	0.9996	0.0012	0.0031	0.0093	0.9938
C 3	245.0	2571.0	2.0	2.0	0.9986	0.9919	0.9992	0.9919	0.9992	0.0008	0.0081	0.0081	0.9919
C 4	163.0	2654.0	3.0	0.0	0.9989	1.0	0.9989	0.9819	1.0	0.0011	0.0	0.0181	0.9909
C 5	19.0	2800.0	0.0	1.0	0.9996	0.95	1.0	1.0	0.9996	0.0	0.05	0.0	0.9744
C 6	242.0	2576.0	0.0	2.0	0.9993	0.9918	1.0	1.0	0.9992	0.0	0.0082	0.0	0.9959
C 7	86.0	2731.0	1.0	2.0	0.9989	0.9773	0.9996	0.9885	0.9993	0.0004	0.0227	0.0115	0.9829
C 8	355.0	2461.0	2.0	2.0	0.9986	0.9944	0.9992	0.9944	0.9992	0.0008	0.0056	0.0056	0.9944
C 9	121.0	2698.0	0.0	1.0	0.9996	0.9918	1.0	1.0	0.9996	0.0	0.0082	0.0	0.9959
C 10	454.0	2364.0	2.0	0.0	0.9993	1.0	0.9992	0.9956	1.0	0.0008	0.0	0.0044	0.9978
C 11	182.0	2637.0	0.0	1.0	0.9996	0.9945	1.0	1.0	0.9996	0.0	0.0055	0.0	0.9973

Ilustración 214: Métricas que reflejan el rendimiento del modelo Alsharif-Plus para CJ1

ACC	0.9943

Ilustración 215: Accuracy de Alsharif-Plus para CJ1

La curva ROC y el AUC obtenidos se puede ver en la Ilustración 216.

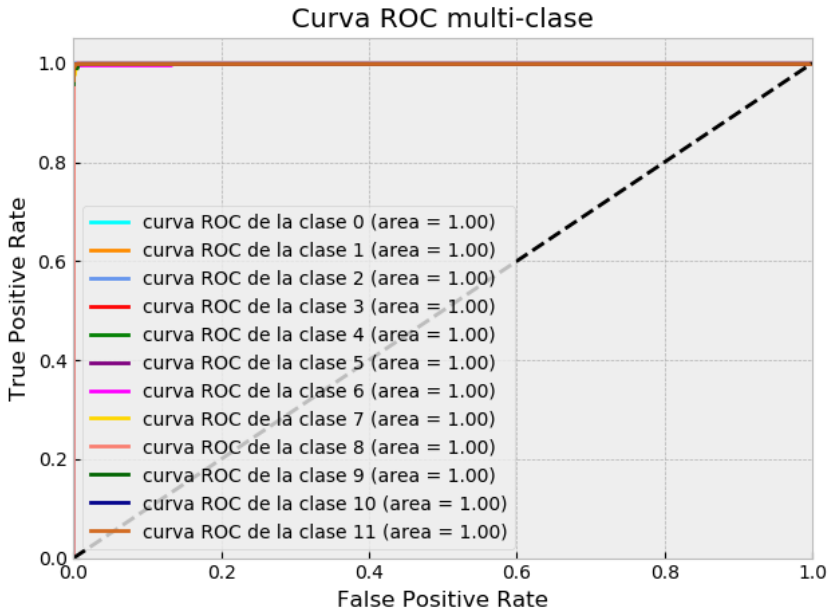

Ilustración 216: Curva ROC y AUC red Alsharif-Plus para CJ1

Por último, observando la Ilustración 217 es donde se afirma dicha mejoría, ya que hay menor número de imágenes mal clasificadas y en este caso la totalidad son imágenes incompletas.

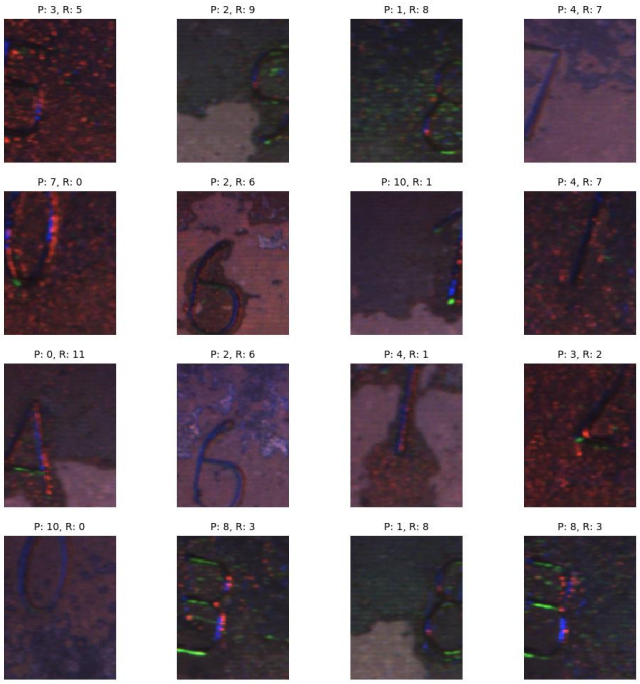

Ilustración 217: Imágenes de CJ1 mal clasificadas por Alsharif-Plus en la etapa de test

- **Conclusiones**
  Tras observar los resultados de la red parece ser que se trata de la que mejores resultados ha arrojado en comparación con las otras dos probadas, obteniendo un ratio de error de **0.56%.**

VI. Conjunto de imágenes CJ2

- **Fase de aprendizaje**
  La evolución del modelo se puede ver en la Ilustración 218, en la Ilustración 219, en la Ilustración 220 y en la Ilustración 221.

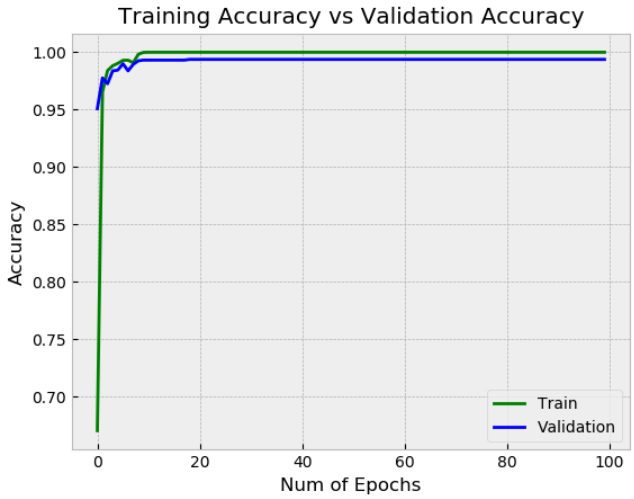

Ilustración 218: Evolución del Accuracy en la fase de aprendizaje con Alsharif-Plus y CJ2

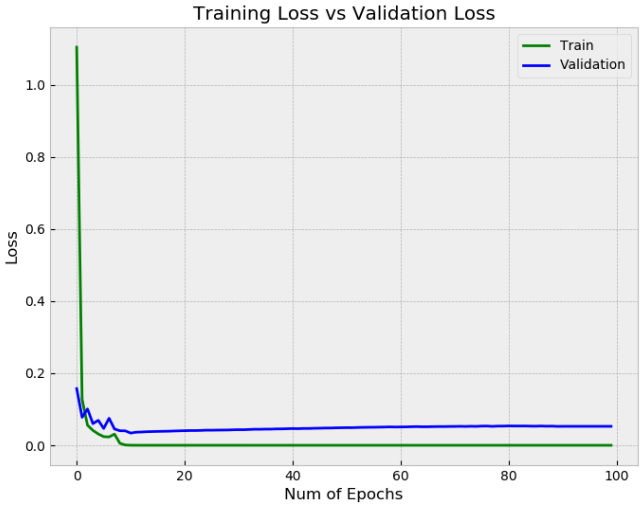

Ilustración 219: Evolución del Loss en la fase de aprendizaje con Alsharif-Plus y CJ2

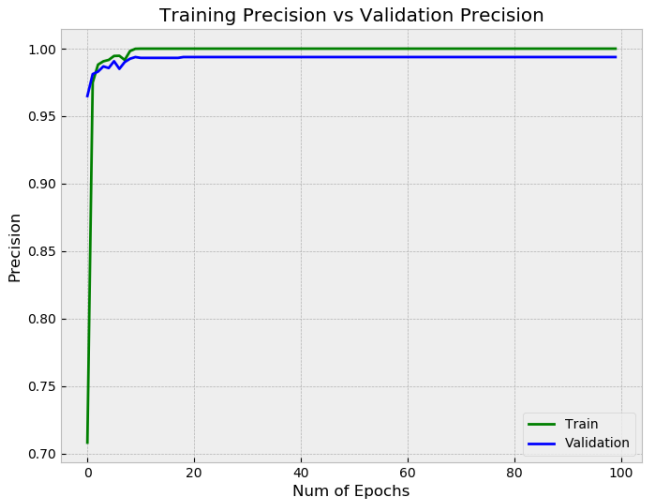

Ilustración 220: Evolución de la Precision en la fase de aprendizaje con Alsharif-Plus y CJ2

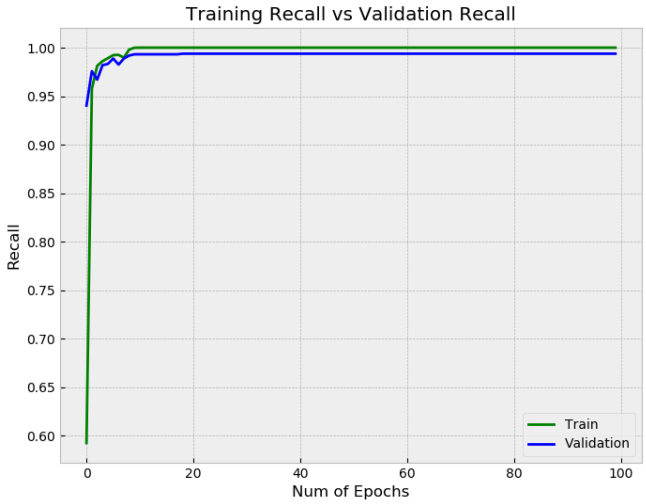

Ilustración 221: Evolución del Recall en la fase de aprendizaje con Alsharif-Plus y CJ2

- **Fase de testing**

  El resultado de testing en forma de matriz de confusión se puede ver en la Ilustración 222. En este caso, se presenta una matriz que refleja mejores resultados de los vistos hasta ahora.

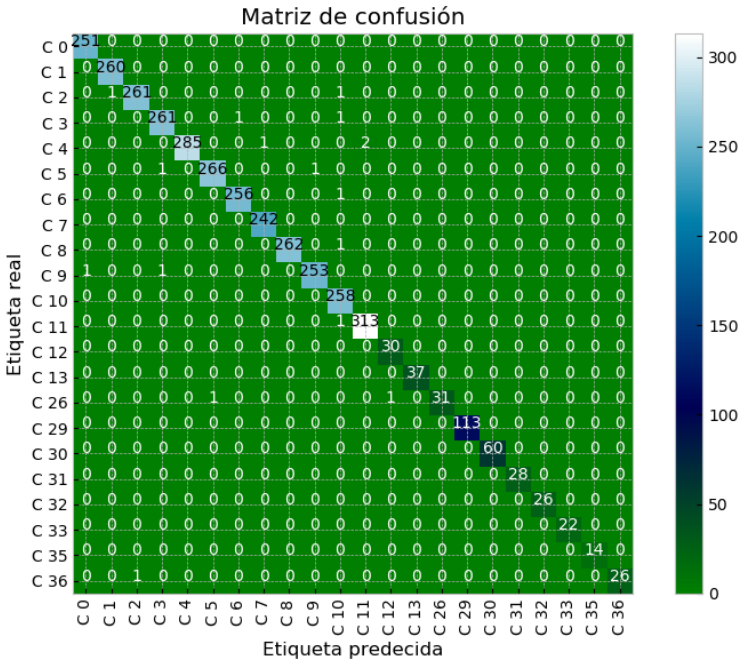

Ilustración 222: Matriz de confusión de la red Alsharif-Plus para CJ2

De este modo, en la Ilustración 223 y en la Ilustración 224, se confirma la situación mostrada por la matriz. Los resultados obtenidos son muy buenos.

	TP	TN	FP	FN	ACC	TPR	TNR	PPV	NPV	FPR	FNR	FDR	F1
C 0	251.0	3320.0	1.0	0.0	0.9997	1.0	0.9997	0.996	1.0	0.0003	0.0	0.004	0.998
C 1	260.0	3311.0	1.0	0.0	0.9997	1.0	0.9997	0.9962	1.0	0.0003	0.0	0.0038	0.9981
C 2	261.0	3308.0	1.0	2.0	0.9992	0.9924	0.9997	0.9962	0.9994	0.0003	0.0076	0.0038	0.9943
C 3	261.0	3307.0	2.0	2.0	0.9989	0.9924	0.9994	0.9924	0.9994	0.0006	0.0076	0.0076	0.9924
C 4	285.0	3284.0	0.0	3.0	0.9992	0.9896	1.0	1.0	0.9991	0.0	0.0104	0.0	0.9948
C 5	266.0	3303.0	1.0	2.0	0.9992	0.9925	0.9997	0.9963	0.9994	0.0003	0.0075	0.0037	0.9944
C 6	256.0	3314.0	1.0	1.0	0.9994	0.9961	0.9997	0.9961	0.9997	0.0003	0.0039	0.0039	0.9961
C 7	242.0	3329.0	1.0	0.0	0.9997	1.0	0.9997	0.9959	1.0	0.0003	0.0	0.0041	0.9979
C 8	262.0	3309.0	0.0	1.0	0.9997	0.9962	1.0	1.0	0.9997	0.0	0.0038	0.0	0.9981
C 9	253.0	3316.0	1.0	2.0	0.9992	0.9922	0.9997	0.9961	0.9994	0.0003	0.0078	0.0039	0.9941
C 10	258.0	3309.0	5.0	0.0	0.9986	1.0	0.9985	0.981	1.0	0.0015	0.0	0.019	0.9904
C 11	313.0	3256.0	2.0	1.0	0.9992	0.9968	0.9994	0.9937	0.9997	0.0006	0.0032	0.0063	0.9952
C 12	30.0	3541.0	1.0	0.0	0.9997	1.0	0.9997	0.9677	1.0	0.0003	0.0	0.0323	0.9836
C 13	37.0	3535.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 26	31.0	3539.0	0.0	2.0	0.9994	0.9394	1.0	1.0	0.9994	0.0	0.0606	0.0	0.9688
C 29	113.0	3459.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 30	60.0	3512.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 31	28.0	3544.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 32	26.0	3546.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 33	22.0	3550.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 35	14.0	3558.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
C 36	26.0	3545.0	0.0	1.0	0.9997	0.963	1.0	1.0	0.9997	0.0	0.037	0.0	0.9811

Ilustración 223: Métricas que reflejan el rendimiento del modelo Alsharif-Plus para CJ2

ACC	0.9952

Ilustración 224: Accuracy de Alsharif-Plus para CJ2

En el caso de la curva ROC y el AUC de la Ilustración 225, el valor obtenido es el máximo posible.

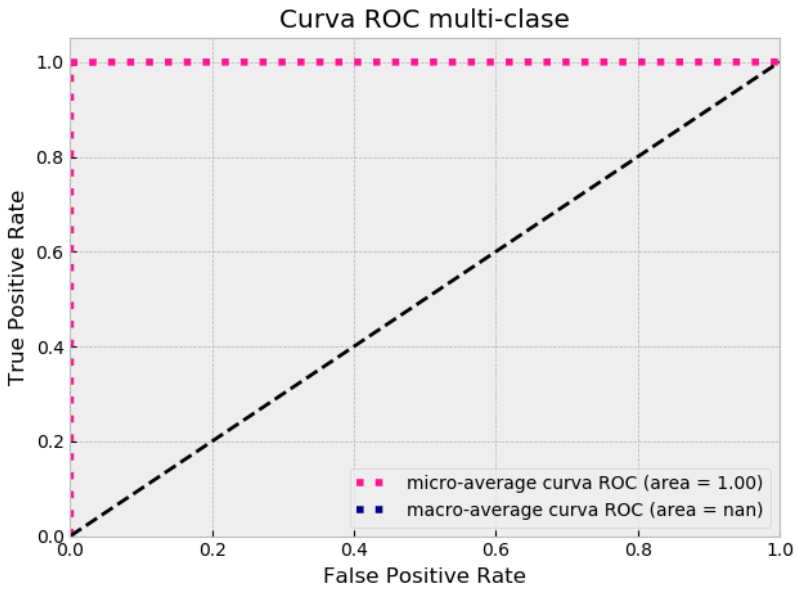

Ilustración 225: Curva ROC y AUC red Alsharif-Plus para CJ2

Finalmente, al igual que ocurrió con CJ1, dicha mejora se puede observar de manera clara en la Ilustración 226. Esto se debe al número de imágenes que contiene, en este caso es menor, y gran parte de estas imágenes se tratan de caracteres incompletos o cuyo fondo no es nítido.

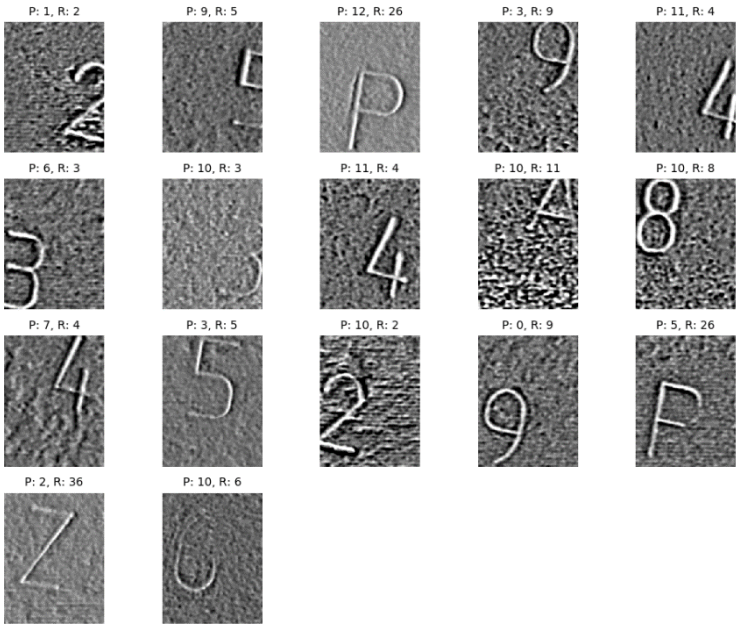

Ilustración 226: Imágenes de CJ2 mal clasificadas por Alsharif-Plus en la etapa de test

- **Conclusiones**
  La red experimentada ha obtenido el mejor rendimiento hasta ahora, teniendo una tasa de error de **0.47%.**

8.2.4 Resultado experimentos finales

Como resultado de la fase final de la experimentación se han obtenido los resultados mostrados en la Tabla 17, obteniendo por una parte las métricas de la última epoch de la fase de aprendizaje y la tasa de error del la etapa de test.

Arquitecturas	*Modelo*	*CJs*	*Accuracy*	*Precision*	*Recall*	*Loss*	*Tasa Error*
LeNet-Plus2	6C5-MP5-16C5-MP5-32C5-MP5-64C5-MP5-F84-10N	CJ1	0.9929	0.9929	0.9929	0.0649	0.70%
		CJ2	0.9924	0.9924	0.9924	0.0817	0.75%
Chen-Plus	16C5- MP3-32C5-MP3-64C3-MP3-128C3-MP3-F1200-10N	CJ1	0.9932	0.9932	0.9932	0.0589	0.67%
		CJ2	0.9904	0.9904	0.9904	0.0841	0.95%
Alsharif-Plus	18C4-MP4-48C8-MP4-128C8-MP4-128C5-MP4-F400-62N	CJ1	0.9943	0.9946	0.9943	0.0628	0.56%
		CJ2	0.9952	0.9955	0.9949	0.0459	0.47%

17: Resultados experimentación redes finales

8.2.5 Solución

Teniendo en cuenta los resultados obtenidos por las redes probadas, se puede ver que el modelo de red que mejor se comporta con los dos conjuntos de datos es Alsharif-Plus, obteniendo el menor ratio de error y los mayores valores para el Accuracy, la Precisión, el Recall y las demás métricas. Motivo por el cual, es la solución elegida para el problema planteado.

9. Metodología de trabajo

A continuación, se realizará una descripción de la metodología de trabajo seguida para llevar a cabo el proyecto. Para ello, se especificarán los recursos utilizados tanto materiales como temporales, así como una división de las tareas realizadas.

9.1 Recursos utilizados

Los recursos utilizados para la realización de este proyecto han sido los siguientes:
Recursos personales: Graduada en Ingeniería Informática de Tecnologías de la Información.
Recursos materiales:

Recurso	*Tipo*	*Finalidad*
Ordenador portátil Lenovo Intel® Core™ i5-6200U, 2,30GHz, 8GB RAM, Windows 10 64 bits	Herramienta de desarrollo	Desarrollo del sistema y la documentación
Servidor	Herramienta de desarrollo	Ejecución de experimentos y parte del desarrollo
JetBrains PyCharm	Framework para desarrollo	Realizar integración de las tecnologías utilizadas
Python	Lenguaje de programación	Implementación del sistema
TensorFlow y Keras	Bibliotecas para Deep Learning	Gestión de CNNs
Halcon MTV	Software de visión artificial	Segmentar imágenes dataset
Microsoft Office Word 2016 Microsoft Office PowerPoint 2016 Microsoft Office Project 2016	Herramientas ofimáticas	Realización de la documentación, planificación y presentación del proyecto

9.2 Desarrollo temporal del trabajo

El proyecto tiene una duración estima de 6 meses y medio, siendo la fecha de inicio el 12/07/2018 y la fecha de finalización 10/02/2019. El tiempo dedicado a su realización inicialmente estuvo ligado a la realización de las prácticas de empresa del Máster, siguiendo un horario de lunes a jueves de 8:00-13:00 y de 14:00-16:15, y los viernes de 8:00-14:00. Tras la finalización de las prácticas se continuo el trabajo con un horario variable.

De este modo, el esquema de planificación llevado a cabo en el desempeño del trabajo se puede observar en la Ilustración 227.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Etapa 1: Introducción al contexto del problema**	**19 días**	**jue 12/07/18**	**vie 10/08/18**	
10	**Etapa 2: Estudio teórico**	**24 días**	**vie 10/08/18**	**mié 19/09/18**	**1**
16	**Etapa 3: Proceso de etiquetado**	**12 días**	**mié 19/09/18**	**mar 09/10/18**	**10**
21	**Etapa 4: implementación del sistema**	**28 días**	**mar 09/10/18**	**mié 21/11/18**	**16**
32	**Etapa 5: Plan de experimentación**	**23 días**	**jue 22/11/18**	**vie 28/12/18**	**21**
36	**Etapa continua: Documentación del proceso realizado en la memoria del trabajo**	**127,25 días**	**lun 16/07/18**	**vie 01/02/19**	

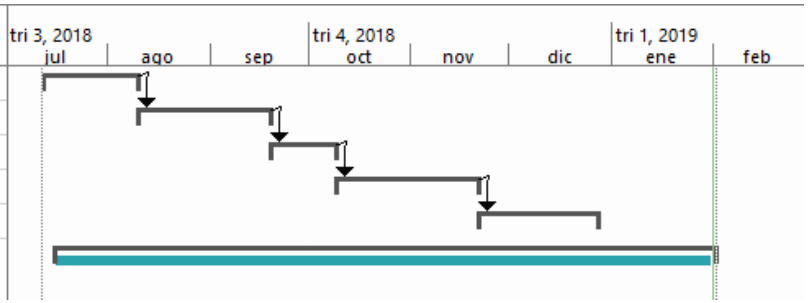

Ilustración 227: Planificación general del trabajo

Como se puede observar, el trabajo fue realizado en diversas etapas, facilitando de este modo el desarrollo del mismo y clarificando las tareas llevados a cabo y que aquellas que aún estaban por realizarse, todo mientras se realizaba la documentación de manera continuada. A continuación, se realizará una descripción breve de cada etapa desglosando las tareas realizadas en cada una de ellas.

- **Etapa 1: Introducción al contexto del problema**
  Se trata de la primera toma de contacto con el problema planteado. Las tareas realizadas se basan en la búsqueda y aprendizaje de conceptos teóricos y prácticos sobre los cuales se iba a trabajar en tareas futuras, además se realizó la delimitación del alcance del problema. El desglose de las tareas llevadas a cabo en esta etapa se puede ver en la Ilustración 228.

- **Etapa 2: Estudio teórico**
  Tras adquirir los conceptos necesarios para poder entender el problema y las posibles soluciones, se realizó un estudio teórico barajando las posibles alternativas de arquitecturas a utilizar, realizando así, una primera elección del conjunto de redes a probar. El desglose de las tareas realizadas se puede ver en la Ilustración 228.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Etapa 1: Introducción al contexto del problema**	**19 días**	**jue 12/07/18**	**vie 10/08/18**	
2	Estudio teórico sobre las CNNs	18 días	vie 13/07/18	vie 10/08/18	
3	**Búsqueda sobre los conceptos base del problema**	**11 días**	**jue 12/07/18**	**lun 30/07/18**	
4	Estudio de la situación actual del sistema	3 días	jue 12/07/18	mar 17/07/18	
5	Introducción conceptos básico sobre visión artificial	3 días	mar 17/07/18	lun 23/07/18	4
6	Introducción a conceptos y desarrollo de TensorFlow	5 días	lun 23/07/18	lun 30/07/18	4;5
7	Elección entorno de desarrollo y puesta a punto de este	2 días	lun 30/07/18	mié 01/08/18	3
8	Realización de pruebas de concepto mediante problemas estandar (MNIST) con Tensorflow y Keras	6 días	jue 02/08/18	vie 10/08/18	7
9	Conclusiones de la etapa de introducción	0 días	vie 10/08/18	vie 10/08/18	8;2
10	**Etapa 2: Estudio teórico**	**24 días**	**vie 10/08/18**	**mié 19/09/18**	**1**
11	Estudio teorico del estado del arte sobre sistemas OCR comerciales	5 días	vie 10/08/18	lun 20/08/18	
12	Prueba del sistema Tesseract	2 días	lun 20/08/18	mié 22/08/18	11
13	Estudio teorico estado del arte de sistemas basados en CNNs para OCR	15 días	mié 22/08/18	lun 17/09/18	12
14	Elección de conjunto de redes a probar	2 días	lun 17/09/18	mié 19/09/18	13
15	Conclusiones estudio teorico	0 días	mié 19/09/18	mié 19/09/18	14

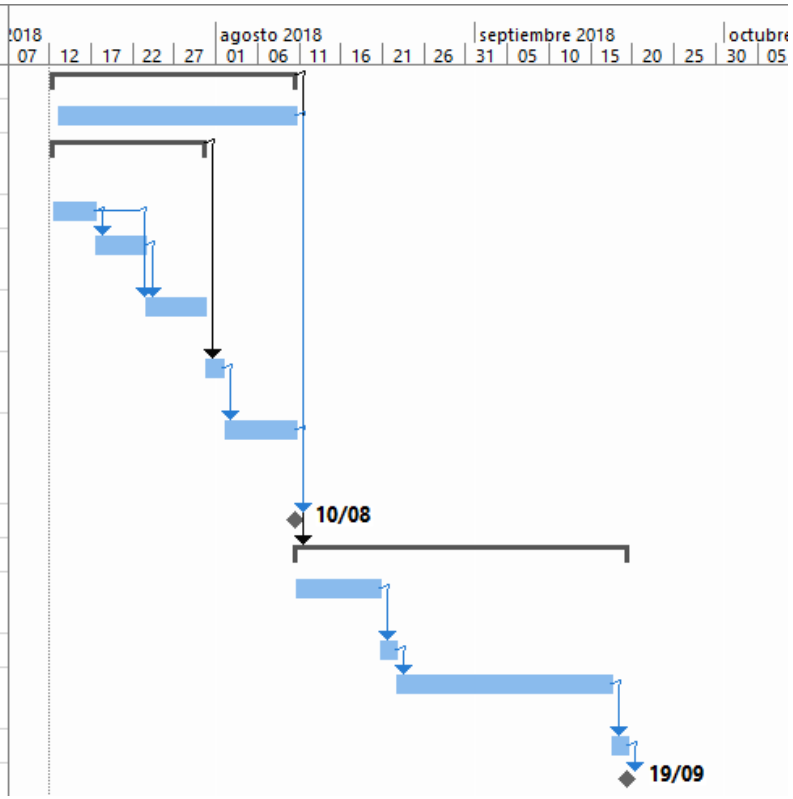

Ilustración 228: Planificación de las tareas realizadas en las etapas 1 y 2 del proyecto

- **Etapa 3: Proceso de etiquetado**
  Se trata de las tareas realizadas para obtener un conjunto de datos que cumpla los requisitos necesarios para ser procesados mediante una CNN de reconocimiento de caracteres. El desglose de las tareas realizadas se puede observar en la Ilustración 229
- **Etapa 4: implementación del sistema**
  Etapa de desarrollo del sistema, en la Ilustración 229 se describen las diferentes tareas realizadas.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
16	**Etapa 3: Proceso de etiquetado**	**12 días**	**mié 19/09/18**	**mar 09/10/18**	**10**
17	Manejo basico Halcon	2 días	mié 19/09/18	vie 21/09/18	
18	Segmentación de imágenes	3 días	vie 21/09/18	mié 26/09/18	17
19	Estiquetado de imágenes recortadas	7 días	jue 27/09/18	mar 09/10/18	18
20	Conjunto de datos etiquetado	0 días	mar 09/10/18	mar 09/10/18	19
21	**Etapa 4: implementación del sistema**	**28 días**	**mar 09/10/18**	**mié 21/11/18**	**16**
22	Carga de datos y generación de archivos TFRecord	5 días	mar 09/10/18	mar 16/10/18	
23	Implementación arquitectura de redes CNN con Keras	2 días	mar 16/10/18	jue 18/10/18	22
24	Desarrollo ciclo completo de trabajo de una CNN	4 días	jue 18/10/18	jue 25/10/18	23
25	Implementación metricas de evaluación y creación de tablas con ellas	6 días	jue 25/10/18	lun 05/11/18	24
26	Creación gráficas	4 días	lun 05/11/18	lun 12/11/18	25
27	Generación modelos resultantes mediante Keras	1 día	lun 12/11/18	mar 13/11/18	26
28	Prueba de funcionamiento del sistema	2 días	mar 13/11/18	jue 15/11/18	27
29	Correción de aspectos valorados en la prueb	3 días	jue 15/11/18	mar 20/11/18	28
30	Prueba final de funcionamiento	1 día	mar 20/11/18	mié 21/11/18	29
31	Sistema preparado para la experimentación	0 días	mié 21/11/18	mié 21/11/18	30

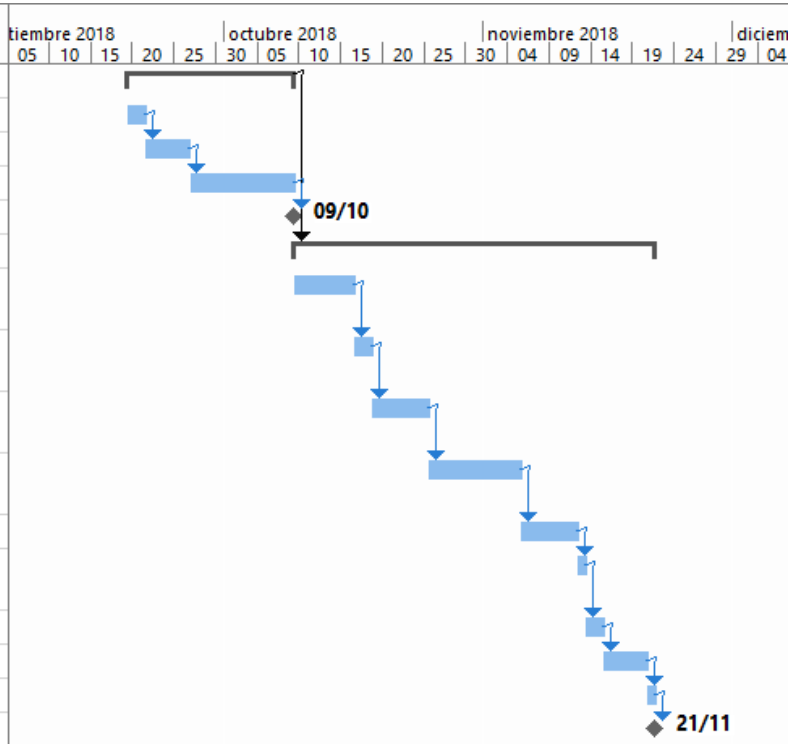

Ilustración 229: Planificación de las tareas realizadas en las etapas 3 y 4 del proyecto

- **Etapa 5: Plan de experimentación**
  Tras el desarrollo del sistema, se continuó con el plan de experimentación para elegir la solución al problema. Dicho plan está basado en el estudio teórico del estado del arte realizado en la etapa 2 del proyecto. El desglose las tareas es el mostrado en la Ilustración 230.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
32	**Etapa 5: Plan de experimentación**	**23 días**	**jue 22/11/18**	**vie 28/12/18**	**21**
33	Experimentación base	14 días	jue 22/11/18	jue 13/12/18	
34	Experimentación final	9 días	jue 13/12/18	vie 28/12/18	33
35	Elección CNN a utilizar como solución al problema	0 días	vie 28/12/18	vie 28/12/18	34

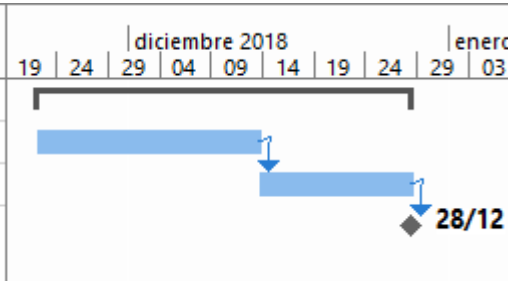

Ilustración 230: Planificación de las tareas realizadas en la etapa 5 del proyecto

- **Etapa continua: Documentación del proceso realizado en la memoria del trabajo**
  De manera paralela a las fases ya descritas, se realizó la tarea de documentación del presente documento, llevando a cabo de manera frecuente varias revisiones del trabajo realizado con el objetivo de asegurar el buen desempeño. Los momentos de dichas revisiones son los mostrados en la Ilustración 231.

Id	Nombre de tarea	Duración	Comienzo	Fin
36	**Etapa continua: Documentación del proceso realizado en la memoria del trabajo**	**127,25 días**	**lun 16/07/18**	**vie 01/02/19**
37	Documentación primera versión de la memori	21,22 días	lun 16/07/18	jue 16/08/18
38	Entrega primera versión de la memoria	0 días	vie 17/08/18	vie 17/08/18
39	Documentación segunda versión de la memor	9,88 días	vie 17/08/18	vie 31/08/18
40	Entrega segunda versión de la memoria	0 días	lun 03/09/18	lun 03/09/18
41	Documentación tercera versión de la memoria	32,53 días	lun 03/09/18	mar 23/10/18
42	Entrega tercera versión de la memoria	0 días	mié 24/10/18	mié 24/10/18
43	Documentación cuarta versión de la memoria	19,41 días	mié 24/10/18	jue 22/11/18
44	Entrega cuarta versión de la memoria	0 días	vie 23/11/18	vie 23/11/18
45	Documentación versión final de la memoria	40,5 días	vie 23/11/18	vie 25/01/19
46	Entrega versión final de la memoria	0 días	lun 28/01/19	lun 28/01/19
47	Revisión versión final de la memoria	2,03 días	jue 31/01/19	vie 01/02/19
48	Entrega memoria final revisada	0 días	vie 01/02/19	vie 01/02/19

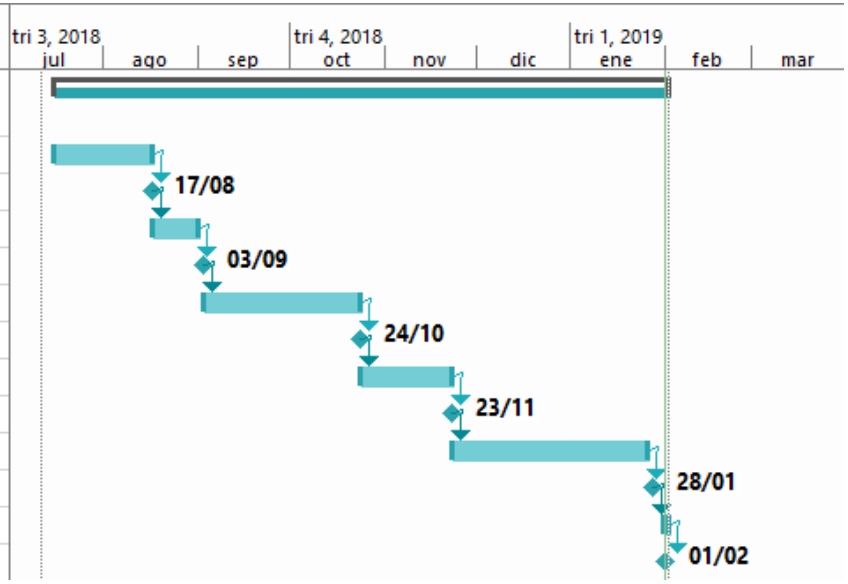

Ilustración 231: Planificación de las tareas realizadas en las etapas de documentación del proyecto

10. Presupuesto

En el presente apartado del documento se describirá el presupuesto de los recursos utilizados para llevar a cabo la realización del trabajo.

10.1 Coste recursos humanos

El desarrollo completo del trabajo fue llevado a cabo por una única Ingeniera Técnica Informática según la planificación vista. El desglose en horas de las tareas realizadas y su correspondiente coste es el mostrado en la Tabla 18.

Etapa	*Horas/etapa*	*Coste/hora*
Etapa 1: Introducción al contexto del problema + Etapa continua: documentación	152	15,00€/h
Etapa 2: Estudio teórico + Etapa continua: documentación	192	
Etapa 3: Proceso de etiquetado + Etapa continua: documentación	96	
Etapa 4: implementación del sistema + Etapa continua: documentación	224	
Etapa 5: Plan de experimentación + Etapa continua: documentación	184	
Etapa continua: documentación final	148	
Total	**996**	**14.940,00€**

Tabla 18: Costes recursos humanos

10.2 Coste recursos materiales

En cuanto a los recursos materiales utilizados para llevar a cabo el desarrollo, se tiene en cuenta el periodo de amortización del hardware, así como las licencias del software utilizado durante el desarrollo, obviándose aquellos cuya licencia es gratuita como son Python y TensorFlow. El desglose de es el mostrado en la Tabla 19.

Recurso	Unidades	Precio unitario	Amortización	Importe
Ordenador portátil Lenovo Intel® Core™ i5-6200U, 2,30GHz, 8GB RAM, Windows 10 64 bits	1	650,00€	25%	162,50€
Servidor	1	3.200,00€	25%	800,00€
Halcon MTV	2			
Office 365	1	69,00€/año	50%	34,50€
Microsoft Office Project 2016	1	25,30€/mes	100%	25,30€
Total				**1.022,30€**

Tabla 19: Costes recursos materiales

10.3 Resumen de costes

De este modo, el resumen del presupuesto utilizado para el desarrollo del proyecto es el mostrado en la Tabla 20. Este presupuesto está formado por el coste del personal y del material utilizado para el desarrollo del proyecto ya que no se han incurrido en otros gastos. Además, se ha utilizado un beneficio empresarial del 15% y el IVA de 21% para obtener el precio final del proyecto.

Recurso	Coste
Recursos humanos	14.940,00€
Recursos materiales	1.022,30€
Total	**15.962,30€**
Beneficio empresarial (15%)	2.394,35€
Total + Beneficio	**18.356,65€**
IVA (21%)	3.854,90
Total + IVA	**22.211,55€**

Tabla 20: Presupuesto final del proyecto

Finalmente, para la realización del proyecto se ha incurrido en un precio de **22.211,55€** con el IVA incluido.

11. Referencias

[Alsharif2013] O. Alsharif, J. Pineau 2013. End-to-End Text Recognition with Hybrid HMM Maxout Models. *ArXiv13101811 Cs*.

[Bai2014] J. Bai, Z. Chen, B. Feng, B. Xu 2014. Image character recognition using deep convolutional neural network learned from different languages. In *2014 IEEE International Conference on Image Processing (ICIP)*,. IEEE: Paris, France; 2560–2564

[Calderon2003] A. Calderon, S. Roa, J. Victorino 2003. Handwritten Digit Recognition using Convolutional Neural Networks and Gabor filters. In *2003 Proceedings of the International Congress on Computational Intelligence CIIC*,. 10

[Chen2015] L. Chen, S. Wang, W. Fan, J. Sun, S. Naoi 2015. Beyond human recognition: A CNN-based framework for handwritten character recognition. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*,. IEEE: Kuala Lumpur, Malaysia; 695–699

[Ciresan2011B] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber 2011. Convolutional Neural Network Committees for Handwritten Character Classification. In *2011 International Conference on Document Analysis and Recognition*,. IEEE: Beijing, China; 1135–1139

[Ciresan2010] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber 2010. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *Neural Comput.*, **22**: 3207–3220.

[Ciresan2011A] D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber 2011. Flexible, High Performance Convolutional Neural Networks for Image Classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*,. 6

[Ciresan2012] D.C. Ciresan, U. Meier, J. Schmidhuber 2012. Multi-column Deep Neural Networks for Image Classification. *ArXiv12022745 Cs*.

[Goodfellow2013] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio 2013. Maxout Networks. *ArXiv13024389 Cs Stat*.

[Hinton2011] G.E. Hinton, A. Krizhevsky, S.D. Wang 2011. Transforming Auto-Encoders. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, Honkela T, Duch W, Girolami M, Kaski S. (eds). Springer Berlin Heidelberg: Berlin, Heidelberg; 44–51

[Ivakhnenko1970] A.G. Ivakhnenko 1970. *PRINTED IN THE U.S.A. Polynomial Theory of Complex Systems PAPER EDITOR'S PREFACE*.

[Jaderberg2014] M. Jaderberg, A. Vedaldi, A. Zisserman 2014. Deep Features for Text Spotting. In *Computer Vision – ECCV 2014*, Fleet D, Pajdla T, Schiele B, Tuytelaars T. (eds). Springer International Publishing: Cham; 512–528

[Kingma2014] D.P. Kingma, J. Ba 2014. Adam: A Method for Stochastic Optimization. *ArXiv14126980 Cs*.

[Krizhevsky2017] A. Krizhevsky, I. Sutskever, G.E. Hinton 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM*, **60**: 84–90.

[Lecun1998] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, **86**: 2278–2324.

[Liu2016] X. Liu, T. Kawanishi, X. Wu, K. Kashino 2016. Scene text recognition with CNN classifier and WFST-based word labeling. In *2016 23rd International Conference on Pattern Recognition (ICPR)*,. IEEE: Cancun; 3999–4004

[Lucas2003] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, R. Young 2003. ICDAR 2003 robust reading competitions. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*,. IEEE Comput. Soc: Edinburgh, UK; 682–687

[Meier2011] U. Meier, D.C. Ciresan, L.M. Gambardella, J. Schmidhuber 2011. Better Digit Recognition with a Committee of Simple Neural Nets. In *2011 International Conference on Document Analysis and Recognition*,. IEEE: Beijing, China; 1250–1254

[Ranzato2007] M. Ranzato, F.J. Huang, Y.-L. Boureau, Y. LeCun 2007. Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*,. IEEE: Minneapolis, MN, USA; 1–8

[Ranzato2006] M. Ranzato, C. Poultney, S. Chopra, Y.L. Cun 2006. Efficient Learning of Sparse Representations with an Energy-Based Model. In *Advances in Neural Information Processing Systems (NIPS 2006)*,. 8

[Rumelhart1986] D.E. Rumelhart, G.E. Hinton, R.J. Williams 1986. Learning Internal Representations by Error Propagation.

[Schmidhuber2015] J. Schmidhuber 2015. Deep Learning in Neural Networks: An Overview. *Neural Netw.*, **61**: 85–117.

[Simard2003] P.Y. Simard, D. Steinkraus, J.C. Platt 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*,. IEEE Comput. Soc: Edinburgh, UK; 958–963

[Srivastava2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res. 15 2014 1929-1958*, 30.

[Wang2011] K. Wang, B. Babenko, S. Belongie 2011. End-to-end scene text recognition. In *2011 International Conference on Computer Vision*,. IEEE: Barcelona, Spain; 1457–1464

12. Anexo I

Sistema de reconocimiento óptico de caracteres (OCR) haciendo uso de un modelo de red neuronal convolucional (CNN)

12.1 Introducción

Estas instrucciones permitirán obtener el proyecto en funcionamiento, pudiendo modificar aquellos aspectos deseados. Y una guía sobre los conceptos más importantes y la estructura de desarrollo.

12.2 Pre-requisitos e instalación

Para poner en funcionamiento el proyecto se necesita crear un entorno adecuado en el que este instalado el lenguaje de programación, biblioteca para ejecutar los modelos, etc.

- Python - El lenguaje de programación utilizado, en este proyecto se utilizó la versión 3.5.0, siendo valida esa o cualquier otra superior.
- Tensorflow - La biblioteca de aprendizaje automático y de código libre más utilizada. Para poder utilizarla se requiere la instalación de otras herramientas. En su web se facilita una guía con las instrucciones a seguir, varían en gran grado dependiendo de si se tiene soporte para GPU o no, para este proyecto es necesaria una máquina con soporte GPU.
- PyCharm - El IDE elegido para realizar el desarrollo, esta elección es libre e independiente para cada desarrollador.

Una vez que se tenga un entorno de trabajo listo en el que se pueda ejecutar Python, se instalarán las siguientes dependencias o paquetes:

- tensoflow-gpu
- Keras
- Pillow
- graphviz (hay que tenerlo instalado previamente)
- imutils
- matplotlib
- numpy
- opencv-python
- pandas, pandas-ml
- scikit-learn
- scipy
- six
- sklearn
- PIL

Todos estos paquetes se pueden instalar desde el IDE elegido, o mediante consola situándose en el proyecto y utilizando el sistema de gestión de paquetes de Python (pip). Un ejemplo:

```
pip install tensorflow-gpu
```

12.3 Estudio teórico previo

Para llevar a cabo el proyecto, inicialmente se realizó una fase de investigación en el campo de las redes neuronales convolucionales para el reconocimiento óptico de caracteres, dando como resultado un amplio estudio, que sirvió para experimentar en busca del mejor resultado. De la gran cantidad de modelos de CNN encontrados en la literatura, se eligió un conjunto reducido, cuyos modelos de CNNs fueron utilizados para realizar el plan de experimentación y tener una base de la solución final que luego se fue modificando. Los artículos se dividen en los que están enfocados solo para reconocimiento de dígitos, y otros basados en el reconocimiento de caracteres alfanuméricos:

Sólo dígitos:

- LeCun 1998: Gradient-Based Learning Applied to Document Recognition
- Ranzato 2006: Efficient Learning of Sparse Representations with an Energy-Based Model
- Simard 2003: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis
- Ranzato 2007: Unsupervised Learning of Invariant Feature Hierarchies with Applications to Objection
- Ciresan 2011: Flexible, High Performance Convolutional Neural Networks for Image Classification
- Meier 2011: Better digit recognition with a committee of simple Neural Nets
- Ciresan 2012: Multi-column Deep Neural Networks for Image Classification
- Chen 2015: Beyond Human Recognition: A CNN-Based Framework for Handwritten Character Recognition

Caracteres alfanuméricos:

- Alsharif 2013: End-to-End Text Recognition with Hybrid HMM Maxout Models
- Jaderberg 2014: Deep Features for Text Spotting
- Bai 2014: Image character recognition using Deep convolutional neural network learned from different languages.

12.4 Estructura del proyecto

El proyecto desarrollado está compuesto de varios directorios donde se encuentran los diferentes "módulos" o "utilidades" para su ejecución. Dicha estructura se hizo con el objetivo de obtener un código más sencillo y fácil de entender. De este modo, la estructura del código es la siguiente:

- Arquitecturas
- Dataset
- Herramientas
- Imagenes
- Preprocesamiento
- Resultados
- StampRail_vX.X.py
- Predecir_modelos.py

La función de cada uno de estos elementos es la siguiente:

Arquitecturas

En este directorio se encuentra el archivo Modelos.py, en cual se han implementado los modelos de CNN basados en el estudio teórico y que se van a probar en el plan de experimentación.

Dataset

En él se almacenan los archivos .tfr creados a partir de los conjuntos de imágenes que conforman un conjunto, por cada conjunto se crean dos, un archivo para la parte de training y validación y otro archivo para el testing.

Herramientas

En este directorio se encuentran varios archivos .py en los que se realizan los procedimientos necesarios en el sistema. Estos son los siguientes:

- Gestion_datos.py: en este archivo se encuentran los métodos encargados de cargar un conjunto de imágenes y sus etiquetas en un array, dividir dicho conjunto en dos subconjuntos, uno de test y otro de trainining con una proporción aleatoria.
- Gestion_TFRecords.py: se encarga de generar los archivos tfrecord y su posterior lectura, obteniendo finalmente las imágenes en un array y su etiqueta en otro, la distribución de los datos es aleatoria.
- Obtener_metricas.py: métodos para calcular las métricas de evaluación (accuracy, recall, precision) y algunos callback del modelo creado por Keras.
- Gestion_graficas.py: Obtener gráficamente la evolución del training, la matriz de confusión, tablas de métricas, imágenes de las capas del modelo, de los pesos de las capas, visualizar las imágenes clasificadas incorrectamente y poder hasta almacenarlas.
- Gestion_modelo.py: funciones para almacenar el modelo de CNN en archivos .JSON, los pesos obtenidos, para poder luego cargar el modelo y realizar predicciones.

Imágenes

En este directorio están almacenados los dos conjuntos de imágenes utilizados para el problema.

- CJ1: Conjunto de datos inicial solo contiene el conjunto de dígitos, imágenes con fondo y la letra 'A'. Se han etiquetado utilizando para los dígitos su propio dígito, para las imágenes de fondo se ha asignado la etiqueta 10 y para la letra 'A' la etiqueta 11. Aquí se encuentran las imágenes en formato .tif organizadas en carpetas por caracteres. Además, las imágenes que se facilitaron en este conjunto estaban nombradas con la siguiente nomenclatura "clase probabilidad tiempo.tif", de modo que para poder cargar los datos en arrays, se tiene en cuenta el separador como " " y la posición 0 donde se encuentra la etiqueta, lo ideal sería que las imágenes siguiesen esa nomenclatura para no tener que modificar código. Por último, hay otro directorio CJ1_evaluar con el que se prueba el modelo generado, en él se encuentran un conjunto de imágenes de todas las clases.
- CJ2: Conjunto de datos ampliado contiene el conjunto de dígitos y varias letras más. Se ha continuado etiquetando de igual forma los dígitos, y ampliando en el caso de las letras las etiquetas de manera consecutiva. Así, si la letra 'A' es etiquetada como 11, se continuará asignando a cada letra sucesiva en el alfabeto un número sucesivo, siendo la última letra 'Z' la etiqueta 36. Aquí se encuentran las imágenes en formato .tif organizadas en carpetas por caracteres. Además, las imágenes que se facilitaron en este conjunto no estaban nombradas de manera que éstas pudieran ser reconocidas por el nombre, por lo tanto se renombraron siguiendo la jerarquía de directorios en las que estaban almacenadas y separándolos por "-", así el nombre de cada imagen se estructura "id-clase-tipoconjunto-tipocaracter.tif", de modo que para poder cargar los datos en arrays, se tiene en cuenta el separador como "-" y la posición 1 donde se encuentra la etiqueta, lo ideal sería que las imágenes siguiesen esa nomenclatura para no tener que modificar código. Por último, Además, hay otro directorio CJ2_evaluar con el que se prueba el modelo generado.

Preprocesamiento

En una fase anterior se encontraban algunos algoritmos de segmentación, ahora hay programas para etiquetar adecuadamente las imágenes y se puedan generar los conjuntos de datos de forma automática utilizando las herramientas ya descritas. En Reetiquetar_imagenes_proc.py se encuentra el código del procedimiento utilizado para reetiquetar el CJ2. En Reetiqueta_A.py, el procedimiento para reetiquetar las imágenes de la clase 11 para el CJ1, que originalmente estaba etiquetado como 'A'.

Resultados

En este directorio se van generando carpetas con los resultados de un experimento, se crean carpetas identificadas con la fecha exacta de generación del experimento del modo "DDMMAAAA-HHMMSS", en cada directorio se almacena un gráfico del modelo, las gráficas de evolución, tablas de métricas, matriz de confusión, historial del entrenamiento y las imágenes mal predichas entre otras cosas.

StampRail_vX.X.py

Se trata del main del sistema, en él se llaman a todos las herramientas y métodos desarrollados. Es decir, se cargan los datos, se compila el modelo, realiza la fase de entrenamiento, el testing, se generan las gráficas y las tablas de los resultados, genera el modelo y luego lo prueba. En la versión 1.2 se realiza este procedimiento para el dataset CJ1 y en la versión 1.3 pare CJ2, ya que difiere el problema en cuanto a número de clases a predecir. La versión 1.1, es una versión inicial que fue mejorada en el 1.2.
En la primera parte de estos archivos, se encuentra una parte de configuración del experimento. Es decir, donde se encuentran los directorios de datos y resultados, definición de número de clases, dimensión de las imágenes, epochs, proporción para la división del conjunto en test, train y validación, learning rate, el tamaño del batch utilizado para los datos, nombre de los archivos .tfr que se van a generar, etc. Para la versión 1.3 el segmento de código es el que se puede ver a coninuación.

```
CURRENT_PATH = os.getcwd()
NOM_TRAIN = 'datostratadostrain.tfrecords'
NOM_TEST = 'datostratadostest.tfrecords'
DIR_DATA = CURRENT_PATH + '/Imagenes/CJ2'
DIR_TFRDATA = CURRENT_PATH+ '/Dataset'
DIR_DATATEST = CURRENT_PATH + '/Imagenes/CJ2_evaluar'
DIR_RESULTADOS = CURRENT_PATH + "/Resultados/" + time.strftime("%d%m%Y-
%H%M%S/")
os.makedirs(DIR_RESULTADOS)
NUM_CLASES = 37
NUM_ETQ =22
CLASES = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
'13', '26', '29', '30', '31', '32', '33', '35', '36']
TARGET_NAMES = ['C 0', 'C 1', 'C 2', 'C 3', 'C 4', 'C 5', 'C 6', 'C 7', 'C
8', 'C 9', 'C 10', 'C 11', 'C 12', 'C 13', 'C 26', 'C 29', 'C 30', 'C 31',
'C 32', 'C 33', 'C 35', 'C 36']
METRICAS =['TP', 'TN', 'FP', 'FN', 'ACC', 'TPR','TNR', 'PPV', 'NPV', 'FPR',
'FNR', 'FDR', 'F1']
HEIGHT = 211
WIDTH = 161
DEPTH = 3
EPOCHS = 100
INIT_LR = 1e-3
BS = 32
VAL_SPLIT = 0.15
TAM_TEST = 0.25
```

Otro de los aspectos configurables y cambiables es el modelo que se quiera probar, para ellos se cambia la función fijada en el método .compile, además para probar modelos de CNN no implementados, basta con añadirlo el nuevo modelo en el directorio Arquitecturas/Modelos.py ya descrito. El cambio de modelo se realiza en la línea de código mostrada a continuación.

```
model = mod.LeCun1998(HEIGHT, WIDTH, DEPTH, NUM_CLASES)
```

Se cambiaría LeCun1998() que es un modelo implementado por otro distinto.

Por último, además del modelo de CNN, se puede cambiar el optimizador utilizado, en el código proporcionado se ha probado con dos funciones de optimización distintas, SGD y Adam, pero en caso de agregar alguno más solo se tendría que definir y cambiar en el parámetro pasado a la función .compile como se muestra a continuación.

```
optA = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
optS = SGD(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=optS,
    metrics=['accuracy', metric.recall, metric.precision])
```

Predecir_modelos.py

En caso de solo querer clasificar un conjunto de imágenes nuevas, se utiliza este archivo, en el se carga un modelo y se realiza una predicción basado en este. Habría que configurar de manera adecuada los directorios donde se encuentra el modelo y el conjunto de imágenes, se obtiene una salida por pantalla de los resultados obtenidos.

12.5 Construido con 🛠

- PyCharm - El IDE utilizado
- Python - El lenguaje de programación utilizado
- Tensorflow - Biblioteca para aprendizaje automático
- Keras - API para CNNs