Rendimiento de distintas arquitecturas de redes convolucionales para la clasificación de imágenes de gestualización numérica decimal con una mano

https://github.com/cabellocarlos/keras-cnn

Carlos Manuel Cabello Colmenares
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
carcabcol@alum.us.es y cabellocarlosmanuel@gmail.com

Iván Cárdenas Meneses
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
ivacarmen@alum.us.es y ivan.cardenas@hotmail.es

***Resumen*—El objetivo principal de este trabajo consiste en entrenar una red convolucional para que esta sea capaz de clasificar imagenes de manos (en escala de grises) que están gestualizando números del 0 al 5. El modelo debe procesar una imagen y predecir el número que se está marcando con los dedos. Además, se busca que el modelo sea capaz de realizar la tarea comentada con la máxima generalización posible, es decir, adaptándose a la mayor cantidad posible de nuevos datos.**

Las conclusiones obtenidas en general después de haber probado con diferentes arquitecturas de menor y mayor tamaño, así como haber aplicado técnicas de regularización es el gran valor que podemos obtener del data augmentation a la hora de buscar aumentar la generalización del modelo que un modelo pequeño con los datos por defecto producen demasiado overfitting y esto se resuelve completamente en una red más grande aplicando data augmentation.

Lo poderosa que es la técnica de data augmentation y como se comporta la curva de aprendizaje de la red cuando se aplica data augmentation.

***Palabras clave*—Inteligencia Artificial, Redes Neuronales Convolucionales**

I. Introducción

Las redes neuronales artificiales [16] son métodos computacionales algo inspirados en las conexiones cerebrales (aunque su funcionamiento dista mucho del de las neuronas reales) , los cuales permiten a un modelo aprender y reconfigurarse a partir del procesamiento de un conjunto de datos que se le ofrecen con el objetivo de resolver nuevos problemas relacionados con los datos aportados a este durante el aprendizaje. Este método puede abarcar gran cantidad de problemas y aprender gran cantidad de conceptos.

La idea de la red neuronal es tener conectadas en capas gran cantidad de neuronas que se encarguen de analizar cada detalle para obtener la solución que se le pide. Con esta idea en mente podríamos pensar que el hecho de introducir la imagen separada por el valor de cada pixel y analizar pixel por pixel podría ser la solución que buscamos para clasificarla correctamente. Sin embargo, una red neuronal estándar con todas sus neuronas densamente conectadas no es nada adecuado en el procesamiento de imágenes, ya que aunque nuestro modelo analice pixel por pixel, nosotros necesitamos que "vea" la imagen en conjunto.

Un tipo de red neuronal [17] que se adapta muy bien al procesamiento de imágenes es la red neuronal convolucional, la cual se compone de capas convolucionales que analizan partes de la imagen por "ventanas" con la ayuda de filtros, que luego reducimos, haciendo un recorrido desde detalles simples como puntos o líneas hasta conceptos abstractos como "mano" en nuestro caso.

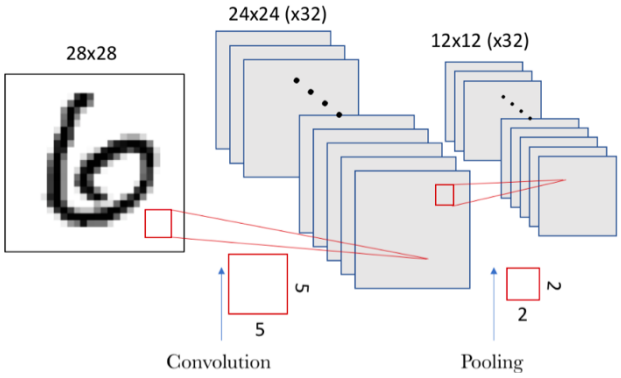


Fig. 1. Muestra de convoluciones

Y por este motivo las redes convolucionales han supuesto grandes avances en la visión artificial en los último años y por tanto son la arquitectura más adecuada para resolver nuestro problema.

El modelo recibe las imágenes de las manos y las procesa en filtros con el uso de ventanas que se fijan en distintas partes de la imagen, siendo capaz de aprender primeramente conceptos como una línea hasta cómo sería una mano gestualizando el número 2 por ejemplo. Teniendo en cuenta esto, el principal problema es encontrar los proporcionar los datos adecuados y analizarlos con la estructura más adecuada para ser capaz de clasificar el mayor número de manos posible y reducir el número que representan correctamente.

Un problema del que nos hemos percatado es que el dataset ofrecido para entrenar nuestros modelos es demasiado

homogéneo, todas las imágenes se parecen demasiado incluso en las imagenes dedicadas a realizar test se observa demasiada dependencia con el caso de entrenamiento. Esto nos ha llevado a alterar el set de datos con el objetivo de tener mayor diversidad en estos. Además nos hemos enfocado en ver si para este problema tan concreto era mejor tener pocas capas o si por otra parte sería más adecuado disponer de más convoluciones para delimitar de forma más precisa el problema , que en este caso concreto buscamos delimitar adecuadamente y de forma generalizada el área de una mano que representa un determinado numero del 0 al 5.

II. Preliminares

Para crear este modelo se han usado arquitecturas basadas en redes neuronales convolucionales, las cuales son muy adecuadas para el procesamiento de imágenes. Dentro de este modelo es necesario realizar otros métodos para garantizar el aprendizaje del modelo y la predicción de datos.

A. Métodos empleados

Describir aquí los métodos y técnicas empleadas (búsqueda en espacio de estados, algoritmos genéticos, redes bayesianas, técnicas de clasificación, redes neuronales, etc.). Si es necesario, separarlos en distintas subsecciones dentro de Antecedentes.

Se pueden usar listas por puntos como sigue:

- Redes convolucionales [13]
- Función de activación ReLu [21]
- Clasificador probabilístico Softmax [22]
- Optimizador de descenso del gradiente estocástico [28]

Por último, se debe hacer un uso correcto de las referencias bibliográficas, para que el lector pueda acceder a más información [**?**]. Todas las referencias al final del documento deben ser citadas al menos una vez.

B. Trabajo Relacionado

Algunos de los trabajos que nos han ayudado a orientarnos con respecto a la arquitectura de nuestros modelos en cuanto al tratamiento de las capas y los hiperparametros son los relacionados con el dataset ImageNet y las competiciones ILSVRC [26] , encontrando modelos muy importantes como VGG16, cuya estructura se detalla en el paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [19] o la anterior a esta AlexNet, detallada en el paper "ImageNet Classification with Deep Convolutional Neural Networks" [27].

III. Metodología

Con el objetivo de hacer un análisis comparativo entre distintas redes convolucionales y su rendimiento, se proponen 2 arquitecturas distintas que se han nombrado como los respectivos autores de cada una de ellas: CarlosNet e IvanNet.

Para evaluar el rendimiento de estas redes se las ha entrenado desde el principio 2 veces:

1) Con el Dataset base
2) Con un Dataset aumentado

A. Datasets

El conjunto de datos sobre el que se va a trabajar como base es el proporcionado por el profesor Agustín Riscos en la asignatura de Inteligencia Artificial. Este dataset contiene un total de 21600 imágenes que a su vez están divididas en los 3 subconjuntos usuales para evaluar un modelo:

- Entrenamiento: 4/6 del conjunto, unas 14400 imágenes.
- Validación:1/6 del conjunto, unas 3600 imágenes.
- Prueba: 1/6 del conjunto, unas 3600 imágenes.

Derivado de este dataset, se ha obtenido un segundo conjunto que introduce más variaciones respecto del original. La diferencia entre los distintos datasets se detalla a continuación.

1) Dataset base: Se compone de imágenes de tamaño 128x128 y en blanco y negro. Las imágenes contienen números del 0 al 5 representados con los dedos de una mano, siempre mostrando la parte de las palmas (nunca el dorso).

Este conjunto de imágenes es especialmente homogéneo como se puede apreciar en la FiguraN. Las manos son aproximadamente del mismo tamaño, se encuentran centradas y contrastan muy bien con el fondo oscuro, siendo la mano mucho más clara. Por otra parte, la orientación de las palmas de las manos varía escasamente y la orientación de las manos siempre es vertical.

2) Dataset aumentado: Partiendo del dataset base, éste se transforma mediante la técnica de data augmentation para conseguir una diversidad de éste mucho mayor. Se ha utilizado la clase ImageDataGenerator de Keras con los siguientes parámetros:

- horizontal_flip=True
- vertical_flip=True
- rotation_range=90
- width_shift_range=0.2
- height_shift_range=0.2
- zoom_range=0.2

De forma aleatoria y para cada batch que se introduzca en el conjunto de entrenamiento, se aplicará una combinación de las transformaciones: Espejo vertical y/o horizontal, rotación, desplazamiento vertical y/o horizontal, ampliación y/ó reducción. Se han escogido estos valores para evitar en lo posible que la mano no se salga de los bordes de la imagen. Como se aprecia en la figura tenemos un conjunto de datos mucho más heterogéneo y que nos permite, en principio, una mejor generalización.

B. Arquitectura de las redes

Las redes se presentan para solucionar problemas parecidos Entrada: Tensor de 128x128x1, por tanto, la imagen debe estar en escala de grises. Salida: Vector de 6 valores [0.,1.] que representan la posibilidad de pertenencia a las respectivas clases.

Para el diseño de la arquitectura de las redes se ha seguido a grandes rasgos los principios heurísticos enunciados en este artículo de towards data science [14] que habla al respecto de la adecuación de los hiperparametros.

1) El tamaño de los filtros debe ser pequeño al principio para conseguir la mayor cantidad de información local posible, para después aumentar gradualmente el tamaño de éstos y extraer características más generales de la imagen.
2) La profundidad de las convoluciones debe ser baja al principio para detectar detalles de bajo nivel, y luego aumentar con el fin de extraer características, formas y patrones más complejos de la imagen.

Con esto, se consigue que las capas iniciales se especialicen en características simples y localizadas de la imagen. Posteriormente, las capas más profundas pueden centrarse en encontrar patrones y conceptos más abstractos, esto se traduce en ir aumentando progresivamente tanto el tamaño de ventana como el número de filtros conforme se avanza en la red.

1) CarlosNet: El diseño de la arquitectura de esta red se ha enfocado en intentar diseñar una red lo más sencilla y con el menor número de parámetros posibles que permitiese un rendimiento óptimo para el dataset que se ha usado en este trabajo. Se ha optado por este enfoque porque, para los estándares de este campo, el tamaño del dataset es excepcionalmente homogéneo y no es demasiado grande. La inspiración para esta arquitectura ha venido de la red LeNet-5 (referencia) y por tanto tiene una arquitectura clásica de red convolucional.

La estructura de la red es muy básica, se alternan capas de convolución y max-pooling 3 veces y luego se hace un aplanamiento para terminar en una última capa neuronal densa con 6 salidas.

1) Entrada: Tensor de 128x128x1.
2) Convolución1: Ventana de 3x3 con 4 filtros.
3) Max-Pooling1: Ventana de 3x3.
4) Convolución2: Ventana de 5x5 con 8 filtros.
5) Max-Pooling2: Ventana de 3x3.
6) Convolución3: Ventana de 7x7 con 16 filtros.
7) Max-Pooling3: Ventana de 2x2.
8) Max-Pooling3: Ventana de 2x2.
9) Flatten: 144 parámetros
10) Softmax: 144 entradas
11) Salida: Vector de 6 valores [0.,1.]

Esta arquitectura no es muy compleja, tiene unos 8000 parámetros de entrenamiento por lo que el coste de entrenamiento no es excesivo.

A la hora de entrenar el modelo con el dataset aumentado, se ha optado por una versión del descenso de gradiente estocástico con un momento de nesterov de 0.8. En distintos ensayos, este valor para el momento de nesterov permitía al modelo converger notablemente más rápido, sobretodo al principio.

2) IvanNet: En total, el modelo se compone de 6167166 parámetros entrenables. Como puede verse, la mayor parte de los parámetros se encuentran en las capas densamente conectadas al final de la red y se encargan de tomar las decisiones.

1) Entrada: Tensor de 128x128x1.
2) Convolución1: Ventana de 5x5 con 20 filtros.
3) Convolución2: Ventana de 5x5 con 20 filtros.
4) Max-Pooling1: Ventana de 2x2.
5) Convolución3: Ventana de 3x3 con 40 filtros.
6) Convolución4: Ventana de 3x3 con 40 filtros.
7) Max-Pooling2: Ventana de 2x2.
8) Convolución5: Ventana de 3x3 con 80 filtros.
9) Convolución6: Ventana de 3x3 con 80 filtros.
10) Convolución7: Ventana de 3x3 con 80 filtros.
11) Max-Pooling3: Ventana de 2x2.
12) Flatten: 9680 parámetros
13) Densa1: 9680 entradas
14) Densa1: 600 entradas
15) Softmax: 300 entradas
16) Salida: Vector de 6 valores [0.,1.]

Como vemos en el esquema, en general el modelo se basa en una red convolucional con: 1 capa de entrada, 7 capas convolucionales, 3 capas de pooling, 1 capa flatten, 3 capas densamente conectadas (una de ellas es la de salida).

Las capas convolucionales disponen de una ventana de 5x5 en las dos primeras y 3x3 en las otras 5 capas. En todas ellas el tamaño de desplazamiento de la ventana es de 1 y la funcion de activacion que se utiliza es RELU. Una de las capas convolucionales actúa como capa de entrada, la cual recibe una imagen de 128x128 y 1 de valor de profundidad (el color de la imagen estará en escala de grises) previamente procesada para poder introducirla en el modelo. Las 2 primeras capas convolucionales tienen 20 filtros cada una, las dos siguientes 40 filtros cada una y las tres últimas disponen de 80 filtros. Las 3 capas de pooling disponen de ventanas de 2x2, lo que reduce el tensor de entrada a la mitad. La capa de salida utiliza el algoritmo de activación softmax.

C. Métricas comparativas

Para comparar el rendimiento de estas redes, se ha optado por las siguientes métricas:

1) Evolución, a lo largo de las épocas de entrenamiento, de la precisión del modelo para el subconjunto de entrenamiento y para el subconjunto de validación.
2) Evolución, a lo largo de las épocas de entrenamiento, de la loss del modelo para el subconjunto de entrenamiento y para el subconjunto de validación.
3) Precisión respecto al subconjunto de prueba.
4) Loss respecto al subconjunto de prueba.
5) Precisión respecto al subconjunto de prueba pero esta vez aumentado.
6) Loss respecto al subconjunto final de prueba pero esta vez aumentado.

Estas métricas se tomaron primero para los modelos entrenados con el dataset original y luego para el modelo entrenado con el dataset aumentado. En keras las medidas de precisión y pérdida vienen dadas por las métricas accuracy y loss respectivamente.

D. Comparando la capacidad de generalización

Primero, se contrastará el rendimiento de los modelos respecto a los subconjuntos originales de validación y prueba.

El objetivo de estos modelos no deja de ser que den el mejor rendimiento posible para el dataset que se ha recogido.

Sin embargo, después, se hará una comparación del rendimiento para la versión aumentada del dataset. Esto servirá para dar una idea del nivel de generalización de los distintos modelos. La forma en la que se ha hecho el aumento de datos, permite que las variaciones introducidas sean aleatorias. Esto simula en buena medida la incertidumbre que puede existir cuando el modelo debe clasificar datos que no se ha encontrado antes. Por ello, consideramos que esta fase experimental es muy importante para valorar la importancia de la técnica de data augmentation.

A continuación, un ejemplo de uso de listas numeradas:

1) *Trabajos con dos alumnos:* poner nombre y apellidos completos de cada uno, y correos electrónicos de contacto (a ser posible de la Universidad de Sevilla). El orden de los alumnos se fijará por orden alfabético según los apellidos.
2) *Trabajo con un autor:* cambiar la cabecera de la siguiente manera
   a) *Una sola columna:* solo se debe especificar un alumno.
   b) *Información a añadir:* la misma que la especificada en el punto 1.

IV. Resultados

A. Entorno de ejecución

La fase experimental se ha llevado a cabo en 2 entornos distintos. Por una parte, la red CarlosNet ha sido entrenada en un ordenador con un procesador intel i7 3770K con 16GB de RAM corriendo Ubuntu. No se ha podido utilizar la aceleración por GPU de Tensorflow porque no se disponía de ninguna tarjeta gráfica compatible. Aún así, la red CarlosNet pudo ser entrenada en unos tiempos muy razonables, del orden de 20-30s por época.

Por otro lado la red IvanNet tuvo que ser entrenada en el entorno de Google Collaboratory, con el entorno acelerado por GPU que se proporciona de forma gratuita. La red se entrenaba muy lento sin aceleración de GPU, por lo que la única alternativa este entorno. La red se entrenaba también en tiempos muy razonables, del orden de 20-30s por época.

B. Entrenamiento con el dataset original

En la primera fase de experimentación, se entrenaron los modelos con el dataset tal y como se proporcionan. Durante el entrenamiento, para cada época, se van registrando las métricas de accuracy y loss para los subconjuntos de entrenamiento y validación.

Ambas redes se entrenaron en esta fase con 10 épocas, y no más, porque que ambos modelos han llegado rápidamente al 100% de aciertos tanto para el subconjunto de entrenamiento como para el de validación como se puede ver en las gráficas 2 y 3.

En la tabla I podemos ver las métricas obtenidas de los modelos después de ser entrenados. Vemos como el modelo predice perfectamente todos los ejemplos de los subconjuntos

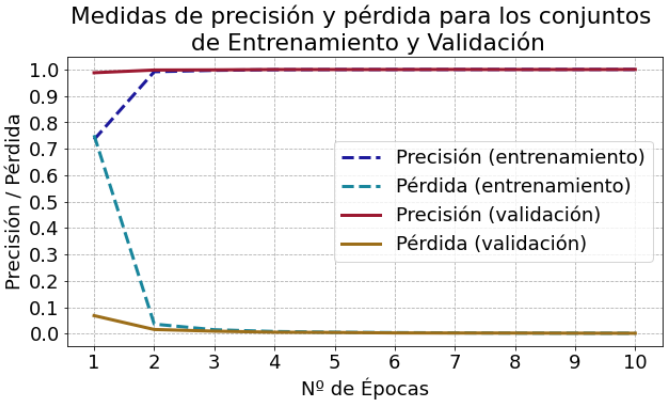


Fig. 2. Curva de aprendizaje para CarlosNet

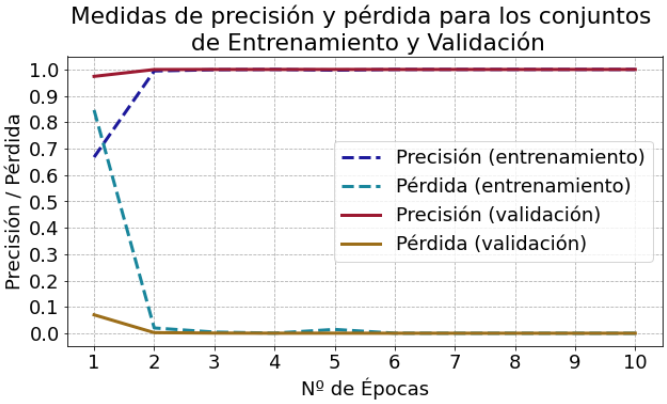


Fig. 3. Curva de aprendizaje para IvanNet

de entrenamiento y validación. Al contrastar los modelos con el conjunto de prueba, IvanNet es capaz de clasificar este conjunto también, siendo CarlosNet la que falla para 2 ó 3 ejemplos.

TABLA I
COMPARATIVA DE LA PRECISIÓN DE LOS MODELOS

	Precisión (train)	Precisión (val)	Precisión (test)
CarlosNet	1.0	1.0	0.9994
IvanNet	1.0	1.0	1.0

En la tabla II podemos ver los valores de pérdida obtenidos al final del entrenamiento, que se mantienen muy consistentes y en magnitudes similares. Sus valores tan bajos nos indican que la función está muy cerca del mínimo global de la función de pérdida, sobretodo IvanNet.

TABLA II
COMPARATIVA DE LA FUNCIÓN DE PÉRDIDA DE LOS MODELOS

	Pérdida (train)	Pérdida (val)	Pérdida (test)
CarlosNet	0.0014	0.0012	0.0031
IvanNet	2.8292e-5	2.076e-5	5.825e-5

La consistencia del buen rendimiento para todos los subconjuntos del dataset nos permiten inferir que ambos modelos se comportan muy bien para datos que sean muy parecidos a los del dataset proporcionado.

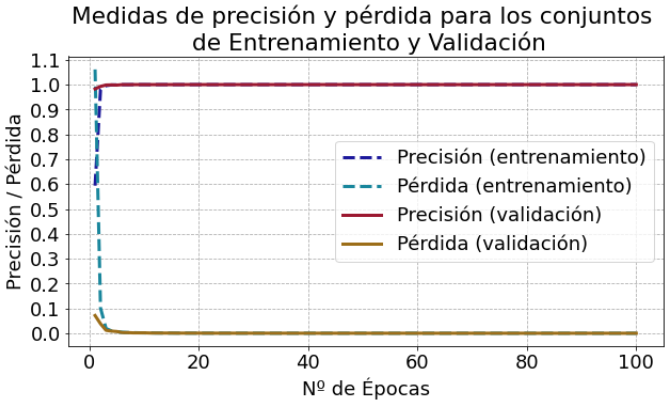
A pesar de que los modelos entrenados parecen funcionar muy bien, se exploró la posibilidad de que esto mismo evidenciaba algo sospechoso respecto al dataset, los modelos o las condiciones de entrenamiento. La rápida convergencia de los modelos puede atribuirse a varios factores:

- A la homogeneidad del dataset.
- A la homogeneidad del dataset.
- Yendo más allá, podrían existir ejemplos repetidos entre los subconjuntos del dataset, dado que no se observa sobreajuste a pesar de la convergencia tan rápida a un 100% de aciertos.

Para descartar que los modelos podrían sufrir de sobreajuste eventualmente, se entrenaron con 100 épocas y tal como se evidencia en las IV-B y 4. Observamos que dado que las gráficas de pérdida y precisión evolucionan casi de forma idéntica tanto para el subconjunto de entrenamiento como para el de validación, no se evidencia ningún signo de sobreajuste. Además, las métricas respecto al conjunto de prueba de la tabla III seguían siendo consistentes con los resultados obtenidos.

TABLA III
COMPARATIVA DE LA FUNCIÓN DE PÉRDIDA DE LOS MODELOS

	Precisión (train)	Pérdida (val)
CarlosNet (100 epocs)	1.0	2.6619e-04
IvanNet (100 epocs)	1.0	3.2875e-04



A priori, no se podría concluir que los modelos o el entrenamiento tienen algún problema, se comportan excepcionalmente bien. Bien es cierto que obtendremos estos mismos resultados si hubiese un solapamiento entre los subconjuntos de validación, prueba y entrenamiento pero analizar la integridad del dataset queda fuera de los límites de este trabajo.

C. Entrenamiento con el dataset aumentado

En la segunda fase de experimentación, se ha entrenado los modelos con el dataset aumentado. Como se ha visto

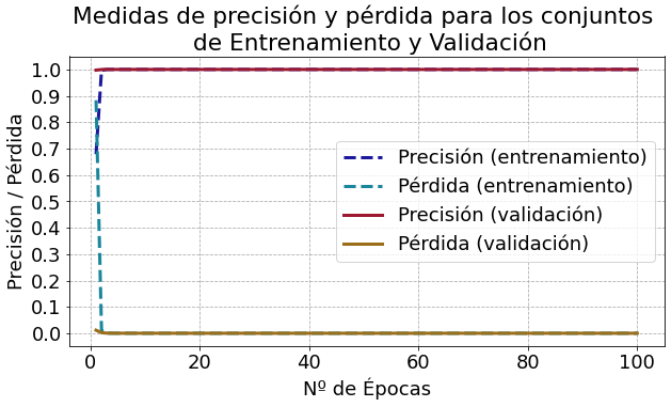


Fig. 4. Curva de aprendizaje para IvanNet 100 épocas

antes, con las técnicas de aumento podemos introducir una heterogeneidad mucho mayor en el dataset, lo que permite que el modelo generalice mejor.

Esta vez, los modelos se entrenan con 30 épocas y como se puede apreciar en las figuras figN y figN los modelos convergen de forma más progresiva. En las gráficas de ambos modelos se evidencian un par de fenómenos interesantes que antes no se producían, nos centraremos en el comportamiento de la función de pérdida:

1) La existencia de picos u oscilaciones de la función de pérdida respecto al conjunto de validación.
2) La función de pérdida del conjunto de validación se mantiene casi siempre por debajo de la función de pérdida el conjunto de entrenamiento.

Se pueden atribuir las siguientes causas a los fenómenos observados[24]:

1) El conjunto de validación no es representativo del dataset, lo que hace difícil evaluar la habilidad del modelo para generalizar.
2) El conjunto de validación es más fácil de predecir que el conjunto de entrenamiento.

El surgimiento de estos fenómenos se debe a que para entrenar al modelo estamos usando el dataset aumentado. Sin embargo, nuestro objetivo sigue siendo que que el modelo dé la mejor predicción posible para el tipo de datos que existe en el dataset tal cual como se proporciona, no para una transformación de este. Por tanto, debemos mantener como referencia el dataset sin aumentar. Como el subconjunto de entrenamiento tiene mucha más diversidad, el subconjunto de validación es más fácil de predecir en comparación, además de que es, en buena medida, diferente al dataset aumentado.

Aún así, se obtienen también muy buenos resultados al final del entrenamiento. Vemos en la tabla IV que el rendimiento respecto al subconjunto aumentado de entrenamiento es del 95% para CarlosNet y de 98% para IvanNet. Es cierto que al compararlo con los subconjuntos de validación y prueba se obtienen peores resultados respecto a los modelos entrenados en la fase anterior, pero la penalización es del orden de 0.4

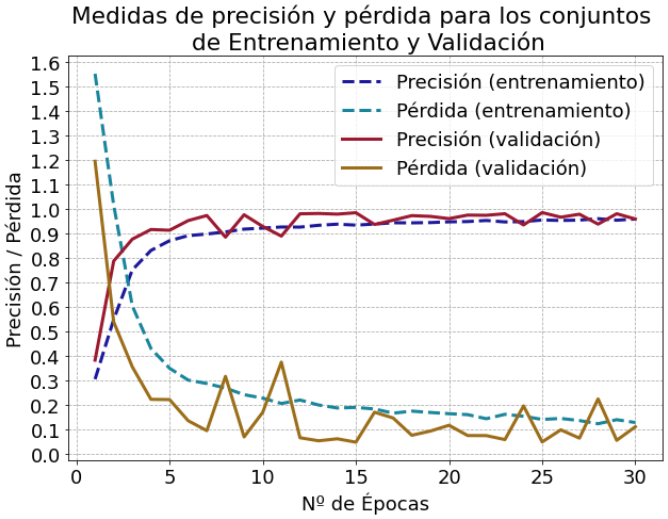


Fig. 5. Curva de aprendizaje para CarlosNet aumentada

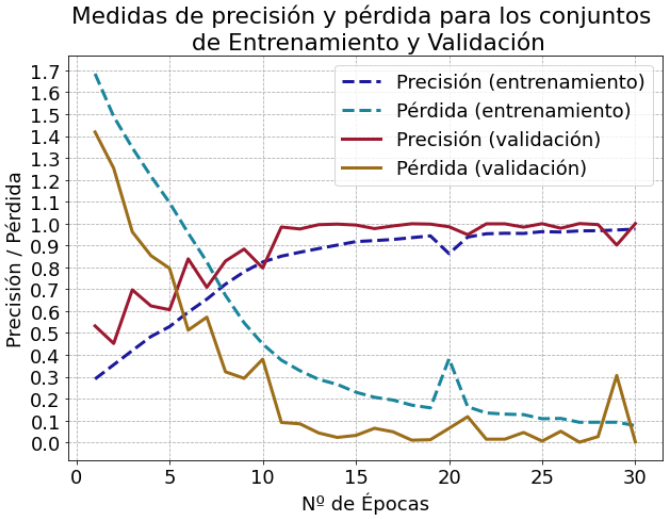


Fig. 6. Curva de aprendizaje para IvanNet aumentada

aproximadamente para CarlosNet y de 0.01 para IvanNet, lo que supone una diferencia relativa bastante pequeña.

TABLA IV
COMPARATIVA DE LA PRECISIÓN DE LOS MODELOS

	Precisión (train_aug)	Precisión (val)	Precisión (test)
CarlosNet_aug	0.9587	0.9586	0.9767
IvanNet_aug	0.9823	0.9942	0.9955

En la tabla V podemos observar como el función de loss también es de varias magnitudes superiores respecto a la fase experimental anterior, pero siguen siendo valores óptimos. Probablemente se podrían conseguir valores más bajos con una mejor función de optimización o más épocas, pero estamos hablando de aumentar el rendimiento de la red en órdenes que no suponen una diferencia sustancial.

TABLA V
COMPARATIVA DE LA FUNCIÓN DE PÉRDIDA DE LOS MODELOS

	Pérdida (train_aug)	Pérdida (val)	Pérdida (test)
CarlosNet (aumentado)	0.1272	0.1099	0.0647
IvanNet (aumentado)	0.0455	0.0017	0.9955

D. Comparación de la capacidad de generalización

Para comparar el nivel de generalización de los distintos modelos, esta vez exponemos a los modelos no entrenados con la técnica de aumento de datos a la versión aumentada del dataset.

Como se evidencia en las tablas [ref] y [ref] el rendimiento de los modelos sin data augmentation esta vez deja mucho que desear, con menos del 20% de aciertos en el caso de ambas arquitecturas de red.

TABLA VI
COMPARATIVA DE LA PRECISIÓN DE LOS MODELOS

	Precisión (train_aug)	Precisión (val_aug)	Precisión (tes
CarlosNet	0.1972	0.1978	0.9767
CarlosNet (aumentado)	0.9587	0.9586	0.9767

TABLA VII
COMPARATIVA DE LA PRECISIÓN DE LOS MODELOS

	Precisión (train_aug)	Precisión (val_aug)	Precisión (test_
IvanNet	0.1981	0.1975	0.1997
IvanNet (aumentado)	0.9823	0.9942	0.9955

Queda de manifiesto que los modelos entrenados con un dataset aumentado incrementan su rendimiento de forma consistente con datos mucho más variados, repercutiendo escasamente en el rendimiento frente al dataset original.

En esta sección se detallarán tanto los experimentos realizados como los resultados conseguidos:

- Los experimentos realizados, indicando razonadamente la configuración empleada, qué se quiere determinar, y como se ha medido.
- Los resultados obtenidos en cada experimento, explicando en cada caso lo que se ha conseguido.
- Análisis de los resultados, haciendo comparativas y obteniendo conclusiones.

V. CONCLUSIONES

Las redes neuronales en general, son una de las técnicas de deep learning más potentes de nuestros tiempos. En este trabajo, hemos profundizado en el funcionamiento específico de las redes neuronales convolucionales, especialmente buenas para enfrentarse a problemas de visión por computador. El surgimiento de APIs como Keras y librerías como Tensorflow han permitido la democratización del conocimiento en estas áreas y han acelerado su innovación. Podemos experimentar fácilmente con estas tecnologías, sin distraernos con detalles

de bajo nivel, lo que nos ha facilitado mucho el desarrollo de este trabajo. Para el problema de clasificación de los dígitos de una mano se han ideado 2 arquitecturas de redes neuronales convolucionales de tamaño muy distinto, así como en su configuración de capas. Hemos comparado su rendimiento al entrenarlas usando la técnica de data augmentation y se ha comprobado lo efectiva que es para mejorar la generalización de los modelos partiendo del mismo dataset.

Una de las conclusiones más evidentes que podemos extraer del desarrollo de este trabajo es lo poderosa que es la técnica de data augmentation para expandir las capacidades de generalización de los modelos a entrenar sin necesidad de recopilar más datos. Hemos visto cómo estas técnicas, al introducir variaciones aleatorias, emulan muy bien las desviaciones que podrían darse una vez que el modelo se exponga a ejemplos que no ha visto antes. Las técnicas de aumento de datos, aplicadas correctamente, apenas suponen una penalización en los modelos a la hora de clasificar los datos del dataset original.

También hemos analizado someramente el comportamiento de las curvas de aprendizaje y qué conclusiones se pueden extraer de éste. Hemos visto que normalmente, al hacer data augmentation y validar con datos del dataset original, obtenemos un comportamiento un poco más errático de las curvas de validación, y que debemos vigilar que al hacer un aumento del dataset no empeoramos el rendimiento del modelo para predicciones parecidas al dataset original.

Una forma mejorar este trabajo es introducir un data augmentation más sofisticado es con la librería imgaug [https://github.com/aleju/imgaug] que permite incluir variaciones en el contraste o el brillo, que pueden ser muy relevantes para generalizar aún más el modelo. Incluye también transformaciones como la inversión del color, la introducción de ruido o el desenfoque. Se podría experimentar también con el nivel de flexibilidad que tienen los modelos entrenados, generando distintos aumentos del datasets con mayot o menor variabilidad.—- Nuestra idea para un futuro trabajo es mejorar mucho más este esquema con la ayuda de datasets mucho más grandes que nos permitan resolver este mismo problema pero con imágenes en color y sin importar el fondo de las imágenes, lo que además permitirá mejorar más aún la generalización de estos modelos.

VI. Anexo: Arquitectura VGG16

VGG16 [18] es un modelo de red convolucional especializado en la clasificación y el reconocimiento de imágenes a gran escala presentada por K. Simonyan y A. Zisserman de la Universidad de Oxford y detallada en el paper “Very Deep Convolutional Networks For Large-Scale Image Recognition” [19] para el ICLR de 2015 [20]. Su nombre es un referencia a las 16 capas de las que se compone este modelo.

La entrada recibida por este modelo es una imagen RGB (3 de profundidad) de 224 x 224 de tamaño. Las imágenes que el modelo es capaz de reconocer y clasificar abarcan hasta las 1000 clases de objetos diferentes.

La arquitectura consta de 13 capas convolucionales y 3 capas densamente conectadas, ordenadas tal y como se ve en la imagen. Las capas convolucionales cuentan con una ventana de convolución de 3x3, la cual es la menor ventana posible para poder representar adecuadamente las nociones de arriba/abajo, derecha/izquierda y centro. Esta ventana se desplaza por los píxeles de la imagen con un desplazamiento de 1. Para evitar perder pixeles en la imagen con cada convolución, se añade un píxel de padding en alto y ancho de la imagen. Las capas max-pooling cuentan con ventanas de 2x2 pixeles de tamaño y un desplazamiento de 2, el cual reduce el tamaño de la imagen a la mitad, en el caso de la VGG, la imagen se reduce 5 veces durante el procesado antes de aplanarse. Los perceptrones de todas las capas ocultas usan la función de activación ReLU [21] para calcular su salida. Dicha función es muy usada en las redes neuronales profundas para visión artificial. Luego de las capas convolucionales se encuentran 3 capas densamente conectadas, es decir, todos los perceptrones de una capa están conectados a las entradas de todos los perceptrones de la capa siguiente. Las dos primeras capas densamente conectadas cuentan con 4096 canales de profundidad cada una y la tercera cuenta con 1000 canales de profundidad (1 canal por cada clase posible) actuando como capa de salida. Esta última capa utiliza una función Softmax [22], la cual se encarga de distribuir la probabilidad de la imagen recibida y procesada entre las 1000 clases posibles, devolviendo un array de valores de 0 a 1 (tipo float) que indican cuán probable es que la imagen pertenezca a dicha clase. Este array es la salida de nuestro modelo. En total, el modelo se compone de 138 millones de parámetros y tiene alrededor de 500MB de espacio de almacenamiento.

Para entrenar este modelo se usó el famoso dataset ImageNet [15], el cual destaca por la competición celebrada anualmente Large Scale Visual Recognition Challenge (ILSVRC). El objetivo de esta competición consiste en enfrentar redes neuronales para clasificar las imágenes de este set de datos y evaluarlas usando una serie de conjuntos de test con los que cuenta esta competición. Concretamente, el dataset que se usó en VGG16 fue el de ILSVRC-2012 (usado desde 2012 hasta 2014). Este dataset incluye imágenes de 1000 clases diferentes y se reparte en tres clases: entrenamiento (1,3 millones de imágenes), validación (50 mil imágenes) y testing (100 mil imágenes). ImageNet cuenta actualmente con más de 14 millones de imágenes distribuidas más de 20000 categorías.

Para el entrenamiento de este modelo se usó el método de descenso del gradiente con momentum. En concreto el momentum estaba al 0.9 y los batch que se introducen, tenían un tamaño de 256. Además, el entrenamiento estaba regularizado por un decaimiento de los pesos, el cual El learning rate inicial es de 0,002 y decrecía en un factor de 10 cuando la precisión con el set de validación dejaba de mejorar . En total, el learning rate decreció 3 veces durante el entrenamiento , parándose después de 73 epoch (370 mil iteraciones).

En las competiciones ILSVRC-2012 y ILSVRC-2013,

VGG16 superó a los anteriores modelos VGG y en el ILSVRC-2013 fue capaz de lograr un 92.7% de precisión en el test de top 5. Además, en el rendimiento de una sola red pudo lograr el 7.0% de error en el test, el mejor resultado, superando por 0.9% a GoogLeNet.

Este modelo demostró que la representación de la profundidad es altamente beneficiosa en la precisión a la hora de clasificar imágenes y que se podía alcanzar el rendimiento state-of-the-art ("estado del arte") [25] de la competición de ImageNet usando una red convolucional convencional.

VII. Referencias

[1]http://www.jetir.org/view?paper=JETIRDA06018

[3]https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

[4]https://machinelearningmastery.com/difference-test-validation-datasets/

[5]https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[6]https://machinelearningmastery.com/image-augmentation-deep-learning-keras/

[7]https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/

[8]https://towardsdatascience.com/convolutional-neural-networks-most-common-architectures-6a2b5d22479d

[9]https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

[10]https://towardsdatascience.com/review-of-lenet-5-how-to-design-the-architecture-of-cnn-8ee92ff760ac

[11]https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7

[12]https://mlfromscratch.com/optimizers-explained/

[13]https://torres.ai/deep-learning-inteligencia-artificial-keras/

[14]https://towardsdatascience.com/a-guide-to-an-efficient-way-to-build-neural-network-architectures-part-ii-hyper-parameter-42efca01e5d7

[15]https://en.wikipedia.org/wiki/ImageNet

[16]https://es.wikipedia.org/wiki/Red_neuronal_artificial

[17]https://es.wikipedia.org/wiki/Redes_neuronales_convolucionales

[18]https://neurohive.io/en/popular-networks/vgg16/

[19]https://arxiv.org/pdf/1409.1556.pdf

[21]https://es.wikipedia.org/wiki/Rectificador_(redes_neuronales)

[22]https://es.wikipedia.org/wiki/Funci%C3%B3n_SoftMax

[23]https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[24]https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[25]https://es.wikipedia.org/wiki/Estado_del_arte

[26]http://www.image-net.org/challenges/LSVRC/

[27]https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[28]https://developers.google.com/machine-learning/crash-course/reducing-loss/stochastic-gradient-descent?hl=es-419