REDES NEURONALES

Resumen

- OBJETIVO:

  Ver cómo de efectivas son las redes neuronales, usando el ejemplo de una base de datos de imágenes de caracteres escritos a mano y tres redes iguales de distinto tamaño.

- MÉTODOS:

  Tenemos 3 redes de diferente tamaño (número de capas). Las entrenaremos, miraremos el tiempo que tardan, su tasa de aciertos y su error (en más detalle en Métodos). Para cada red, usaremos una muestra de 100 de la base de 60.000 imágenes y las testearemos con 10.000 imágenes distintas a las de la base (mirar esquema). De allí sacaremos su tiempo, error y precisión (tasa de aciertos).

- RESULTADOS

  FALTA

- CONCLUSIÓN

  FALTA

Introducción

Ya que estudiamos informática y la inteligencia artificial es una de las ramas más prometedoras del ámbito, hemos decidido centrar nuestro trabajo en las redes neuronales.

Específicamente, hemos escogido estas porque, dentro de la inteligencia artificial y el machine learning, son el método que más eficacia ha demostrado para la gran mayoría de casos en los que podemos aplicar las redes neuronales.

Como concepto básico, las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples, de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas en los cerebros biológicos.

Objetivo

Nuestro objetivo, pues, es determinar la eficacia y eficiencia de las redes neuronales. Concretamente, nos hemos fijado en una, la Deep Convolutional Network.

(Para entender mejor la siguiente parte, es recomendable ver la imagen *Esquema red*, situada en métodos).

Con tres tamaños de red (1, 3 y 5) haremos 5.000 iteraciones en cada, entrenándolas con 100 imágenes por iteración, y testeándolas con 10.000 distintas. Mientras se entrenan también se testean, ya que por cada imagen del entrenamiento, aprende más.

Gracias a eso, llegamos a tener 3 tiempos de entrenamiento, 3x5.000 tasas de aciertos de entrenamiento (una por iteración y red), también 3x5.000 tasas de aciertos del test, y lo mismo para el error.

En resumen, las variables aleatorias que tendremos serán:

Red *i* (5.000 iteraciones)	**(*i* = 1, 3 o 5)**
Entrenamiento: 100 imágenes / iteración	**Test: 10.000 imágenes / iteración**
(x 1) Tiempo de entrenamiento	-
(x 5.000) Precisión entrenamiento	(x 5.000) Precisión test
(x 5.000) Error entrenamiento	(x 5.000) Error test

Métodos

En este apartado explicaremos como hemos llevado a cabo nuestro proyecto.

Redes

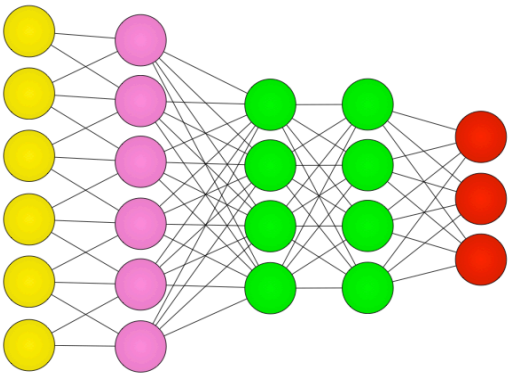
Las tres imágenes siguientes son de *Deep Convolutional Network,* y son las que vamos a estudiar en este proyecto.

Mirándolas, podemos ver que, cuando nos referimos al número de capas, hablamos de “la cantidad de columnas que vemos de color rosa”. Entonces, el hecho de cambiar el número de capas, es solo cambiar esta parte “rosa”, llamada capa *convolutional*.

De esta forma, estamos seguros de que si hay cambio en los resultados, eso es solo porque se ha cambiado esta parte de la red, y ninguna más.

Las 3 redes:

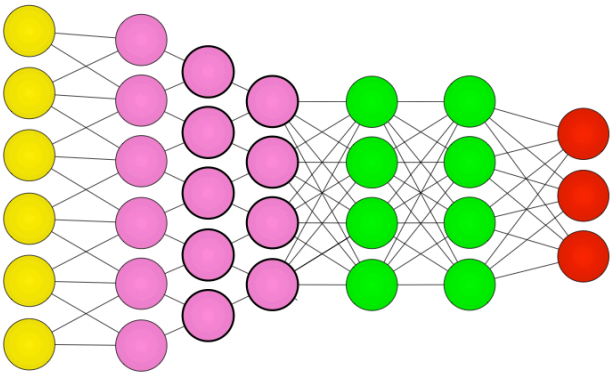
Red neuronal pequeña (*una capa*)



Una capa: Deep Convolutional Network

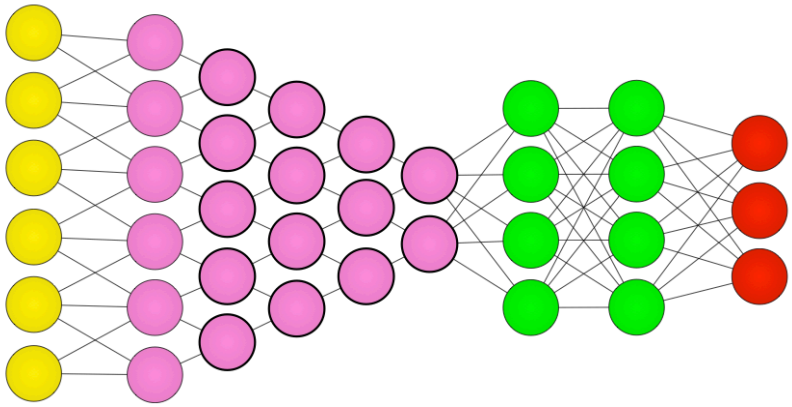
Leyenda

Red neuronal mediana (*tres capas*)



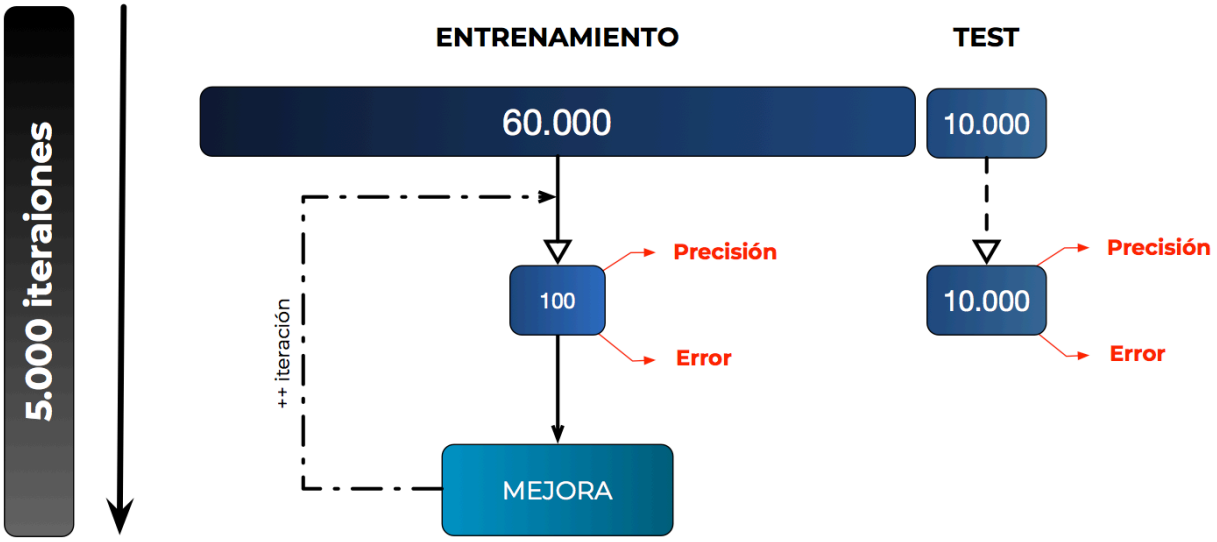
Tres capas: Deep Convolutional Network

Red neuronal grande (*cinco capas*)



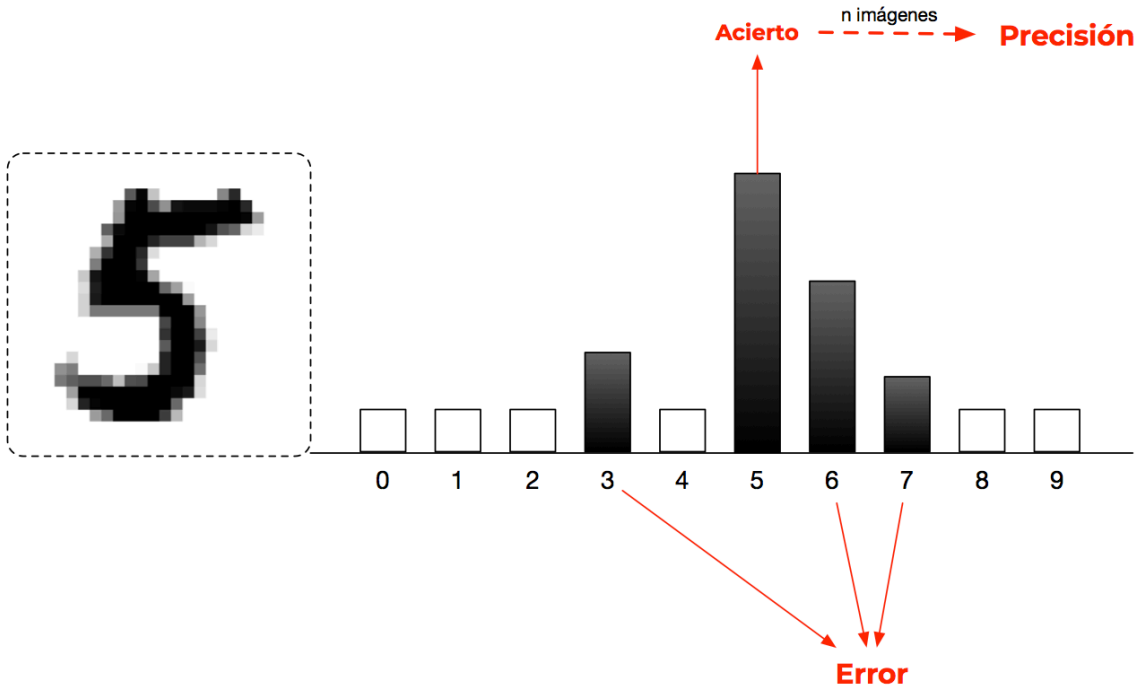
Cinco capas: Deep Convolutional Network

A continuación tenemos el funcionamiento de cada una de las redes (igual para todas):



Esquema red: explicación sobre cómo funciona cada una de las tres redes.

Y también cómo llega la red a calcular la precisión y el error para cada iteración, poniendo el ejemplo de una imagen de la base de datos (60.000):



Esquema ejemplo: para cada imagen, da un seguido de probabilidades para cada carácter que pueda ser. En caso que el que tenga la probabilidad más alta sea el correcto, hay acierto. Con ese acierto y todos los de cada iteración, se calcula la precisión. Las otras probabilidades (en este caso, las de los números 3,6 y 7), hacen que se incremente el error.

Tiempo de entrenamiento (aclaración)

Cada vez que entrenamos una red, se imprime la información de cada iteración. Eso nos suma más tiempo del que realmente tarda la red en entrenarse, ya que tiene que imprimir 5.000 líneas.

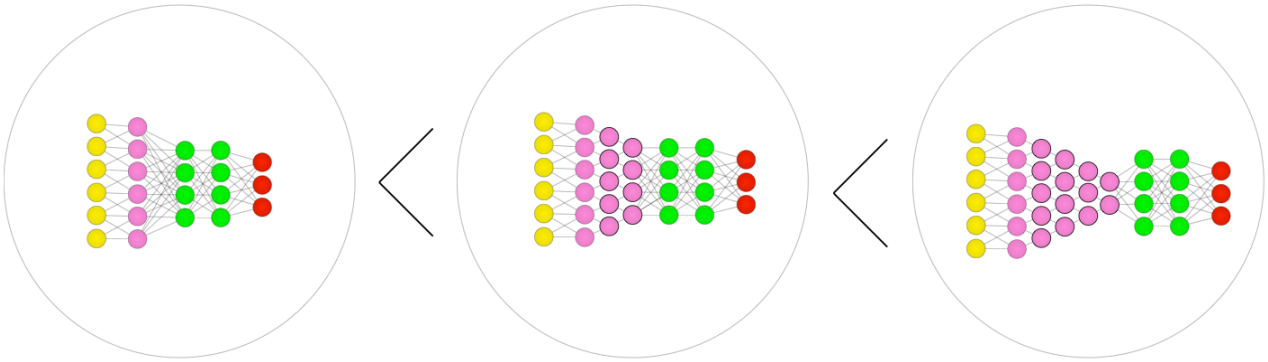
Para solucionar eso, lo que haremos será solo imprimir la primera, y decirle que imprima con un intervalo mayor a 5.000. De esta forma, ahorramos imprimir 4.999 líneas y el tiempo será más próximo a lo que tarda realmente en entrenarse.

Por lo tanto, asumimos que el que nos salga imprimiendo solo una iteración, va a ser el real. Además, como esa impresión se va a hacer en cada red, el “desfase” que nos provoque esta va a ser el mismo siempre y, por lo tanto, no va a afectar a nuestros resultados.

Pruebas de hipótesis

Al tener tres redes, vamos a hacer dos pruebas de hipótesis; en una compararemos la de una capa con la de tres, y en la segunda compararemos la de tres con la de cinco.

Al tener más capas y disponer de los mismos recursos, la red de cinco capas tendría que tener una precisión (en media, es decir, μ) más elevada. Por lo tanto, nuestras hipótesis van a ser:



Pruebas de hipótesis: En media de precisiones, $\mu_1 < \mu_3 < \mu_5$

Prueba para 1 y 3	Prueba para 3 y 5
H_0: $\mu_1 = \mu_3$	H_0: $\mu_3 = \mu_5$
H_1: $\mu_1 < \mu_3$	H_1: $\mu_3 < \mu_5$

Premisas:

- Normalidad: la precisión se distribuye binomialmente, ya que hablamos de aciertos (1 o 0), pero como tenemos una n muy grande (5.000 iteraciones), podemos decir que se distribuye de forma normal.
- Unilateral: nuestra premisa es que, como más capas, mejor es la precisión de la red. Por lo tanto, es unilateral.
- Muestra aleatoria: Es una muestra aleatoria, ya que vamos a coger la precisión del entrenamiento, dónde coge 100 imágenes al azar.

Estadístico

μ	$H_0 : \mu_1 = \mu_2$	$\hat{z} = \frac{(\bar{y}_1 - \bar{y}_2)}{\sqrt{S_1^2/n_1 + S_2^2/n_2}}$	$n_1 , n_2 \geq \approx 100$ m.a.s indep	$\hat{z} \sim N(0,1)$	Rebutjar si $\lvert \hat{z} \rvert > Z_{1-\alpha/2}$

Les corresponents proves unilaterals es fan acumulant el risc α a un costat

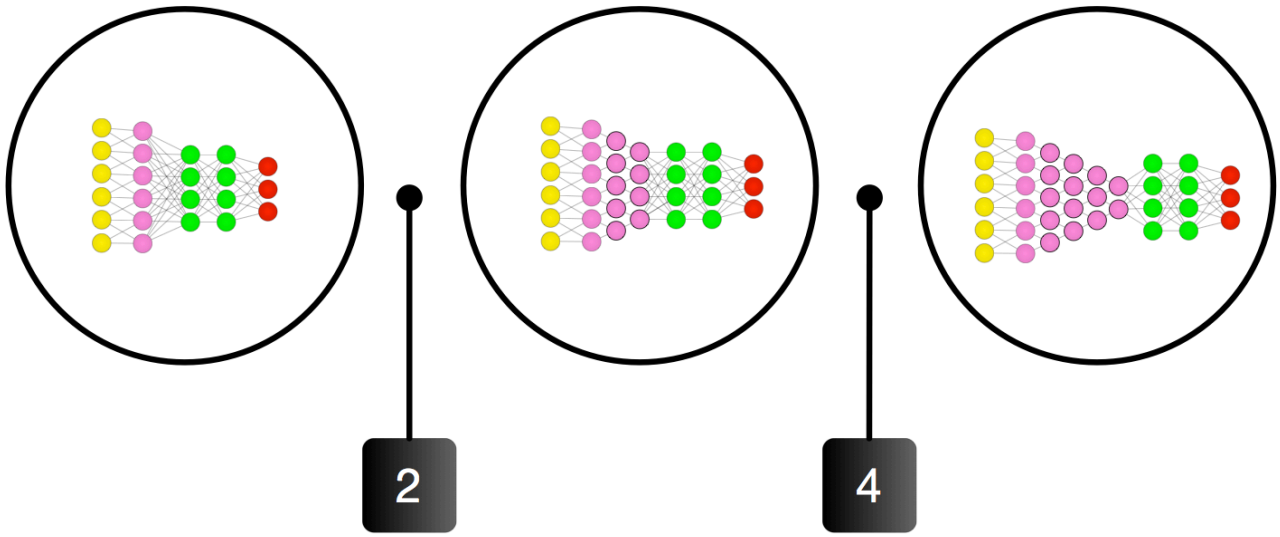
Variables:

- $\alpha = 0.001$ (hemos elegido esta ya que queremos tener mucha confianza).
- Para nuestro estadístico, vamos a usar esta "*z*", ya que nuestras muestras son lo suficientemente grandes ($n_1 = n_3 = n_5 = 5.000 > 100$).
- y_i son las medias de la precisión de entrenamiento de la red_i.
- S_i se calcula de la siguiente forma, donde s_i^2 es el estimador de la varianza de la red_i.

$$s^2 = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}{n-1} = \frac{\sum_{i=1}^{n} x_i^2 - n(\bar{x})^2}{n-1} = \frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}{n-1}$$

Predicción

Para esta parte del proyecto queremos predecir lo que van a tardar otras redes iguales, pero de distintas capas, a entrenarse. Asumiremos que las de 1, 3 y 5 van a formar un gráfico lineal. Trabajando con esta regresión, vamos a poder predecir el tiempo que tardará en entrenarse las redes de tamaños intermedios, de 2 y 4 capas.



Predicción: Tenemos los tiempos de las redes con número de capas = 1, 3 y 5. Con esos datos, haremos una predicción del tiempo que tardarán en entrenarse las redes con números de capas = 2 y 4.

Para estimar los tiempos de entrenamiento hemos ejecutado los programas de las redes neuronales en un mismo ordenador varias veces.

X = Número de Capas
Y = Tiempo (segundos)

X	1	1	1	1	1	1
Y	39.804	39.112	40.261	39.854	31,629	31,645

X	3	3	3	3	3	3
Y	206.42	189.27	189.98	190.19	180.64	177.55

X	5	5	5	5	5	5
Y	436.96	412.55	486.97	508.82	432.21	492.19

A partir de estos resultados haremos una regresión lineal hallando la función que relaciona las capas con el tiempo de entrenamiento. A partir de esta función podremos inferir los tiempos de entrenamiento para aquellas redes neuronales de las cuales no tenemos los datos.

Resultados

Descriptiva

Aquí podemos ver los resúmenes (summary) de las tres redes, que incluyen la precisión y el error, tanto del entrenamiento como del test:

> summary(layer1Data)

Red de una capa	Training.Accuracy	Training.Loss	Test.Accuracy	Test.Loss
Min	0,14000	0,06035	0,14240	5,37500
1st Qu.	0,97000	1,58712	0,97530	5,83200
Median	0,99000	3,58106	0,98020	6,32800
Mean	0,98000	6,76477	0.9733	8,76900
3rd Qu.	1,00000	7,80190	0.9820	7,93800
Max.	1,00000	233,83017	0,98380	232,14900

> summary(layer3Data)

Red de tres capas	Training.Accuracy	Training.Loss	Test.Accuracy	Test.Loss
Min	0,13000	0,00048	0,11350	2,65900
1st Qu.	0,99000	0,17331	0,98650	3,17500
Median	1,00000	0,85188	0,98960	3,39300
Mean	0,98880	3,69085	0,98410	5,12300
3rd Qu.	1,00000	3,73477	0,99050	4,14100
Max.	1,00000	231,18448	0,99200	231,65900

> summary(layer5Data)

Red de cinco capas	Training.Accuracy	Training.Loss	Test.Accuracy	Test.Loss
Min	0,07000	0,00007	0,09310	2,48500
1st Qu.	0,99000	0,02844	0,98820	3,14300
Median	1,00000	0,33070	0,99090	3,33900
Mean	0,99040	3,07318	0,98580	4,85200
3rd Qu.	1,00000	2,43683	0,99220	3,89800
Max.	1,00000	313,25934	0,99350	304,30800

Pruebas de hipótesis (con $\alpha = 0.001$)

- De red pequeña (1) y mediana (3)

  Estadístico: z = -17.12524

  $z_{1-\alpha} = 3.090232$

  Para rechazar: $|z| > z_{1-\alpha}$ (NO α/2 ya que es unilateral)

  Rechazamos H_0 de 1 y 3 -> 3 es mejor que 1, con una confianza del 99.9%

- De red mediana (3) y grande (5)

  Estadístico: z = -3.346447

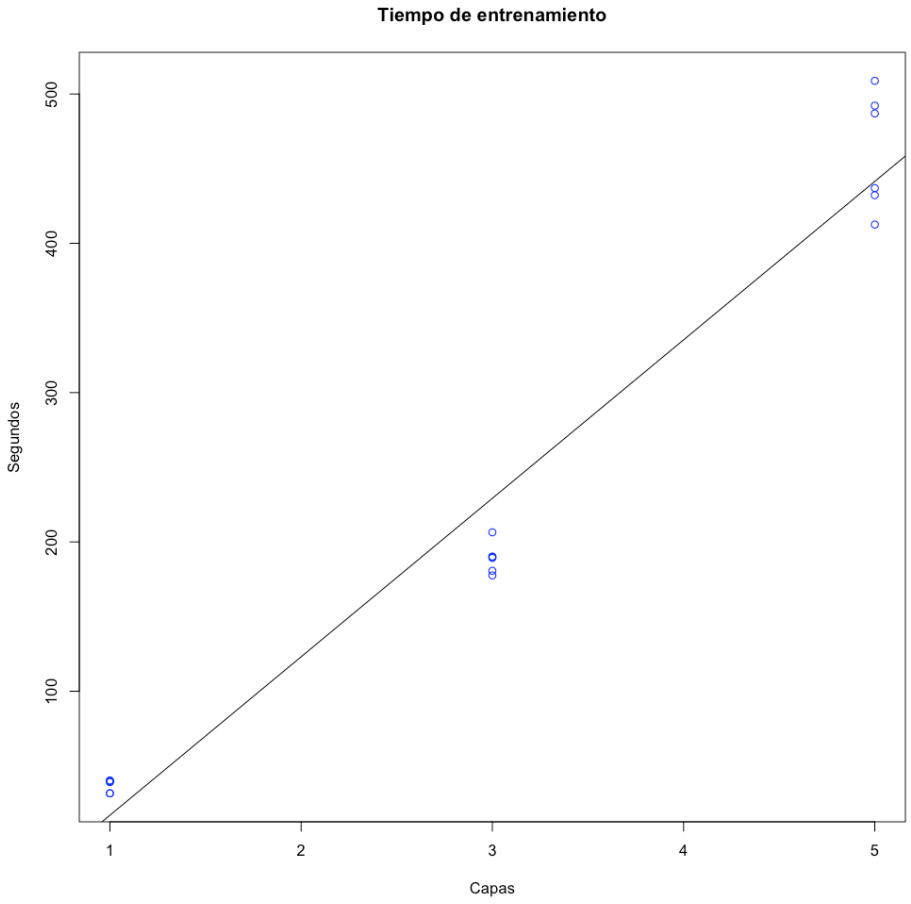
  $z_{1-\alpha} = 3.090232$

  Para rechazar: $|z| > z_{1-\alpha}$ (NO α/2 ya que es unilateral)

  Rechazamos H_0 de 3 y 5 -> 5 es mejor que 3, con una confianza del 99.9%

Predicción

Regresión lineal:



Regresión: función que determina el tiempo de entrenamiento (Y) según el número de capas de la red (X)

FALTA

Discusión

FALTA