Trabajo practico 3
OCR para digitos
Una comparativa entre arboles de decisión y redes neuronales
Introducción a la Inteligencia Artificial

Facundo Emmanuel Messulam y Franco Ignacio Vallejos Vigier

Junio 2022

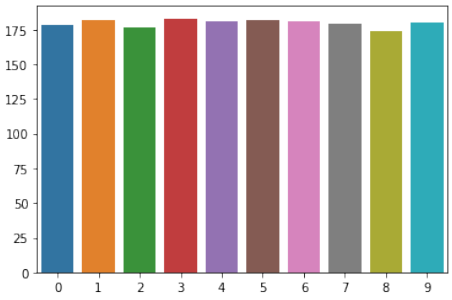
1 Introducción

Se conoce al Optical Character Recognition (OCR) como el sistema que clasifica imágenes de acuerdo a que carácter de un lenguaje representan. En el presente análisis se toma el dataset proveído por la cátedra (analizado con mas detalle en el trabajo), para entrenar un árbol de decisión y una red neuronal convolucional para sacar conclusiones comparativas entre estos dos métodos.

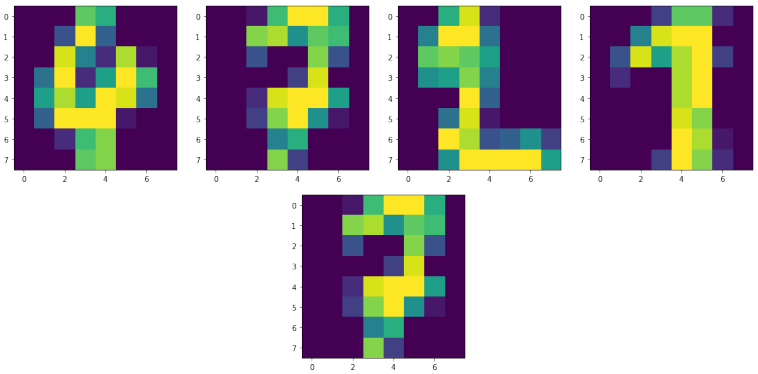
2 Desarrollo

2.1 Dataset

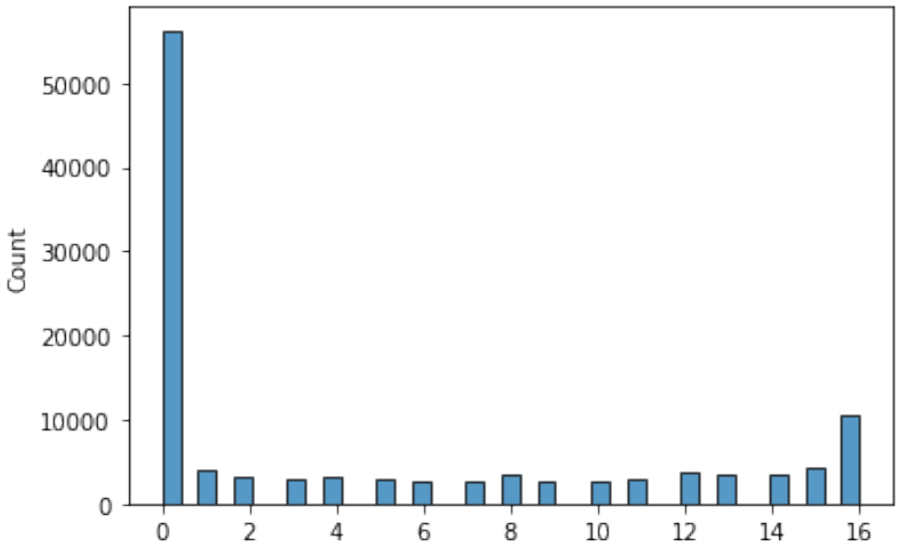
Para empezar se hace un análisis estadístico del dataset. Con una cantidad de imagenes de 1797, y al rededor de 175 imagenes por digito, con clases para cada uno de los 10 digitos.



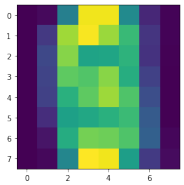
Algunos ejemplos:



La distribución de colores de píxel en las imágenes:



La imagen promedio:



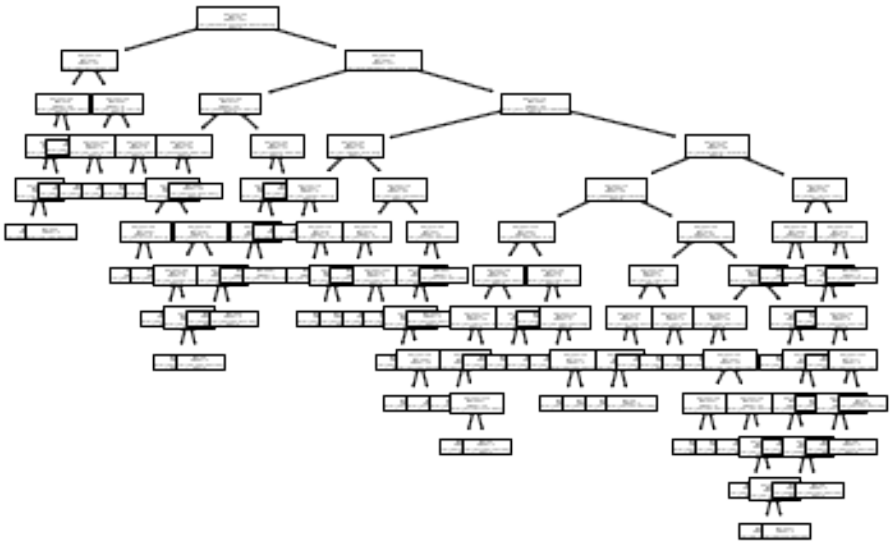
Esto nos dice que los bordes ni se usan y además, la mayoría de las imágenes están alineadas con el carácter dibujado, no hay "ángulos raros" y esto puede llevar a sobreentrenamiento.

2.2 Arboles de decisión

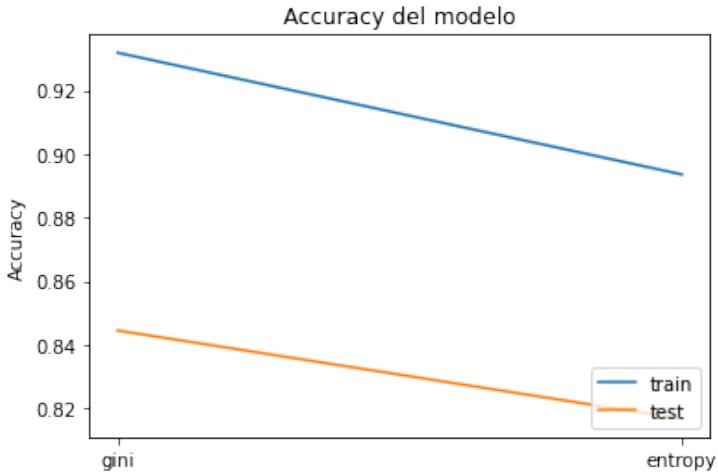
2.2.1 Árbol inicial

Aquí vemos que el entrenamiento produce un árbol extenso, pero la precisión no sube del 86%. Los hiperparamentros elegidos no limitan la extensión del árbol, ni la cantidad de splits, solo establecen que la cantidad de entradas para separar el nodo en dos es de dos, lo que esto genera es amplio sobreentrenamiento donde el set de entrenamiento es perfectamente clasificado, a pesar de que el de validación no lo es.

Para limitar este sobre-entrenamiento se usa pruning, generando un árbol más simple:



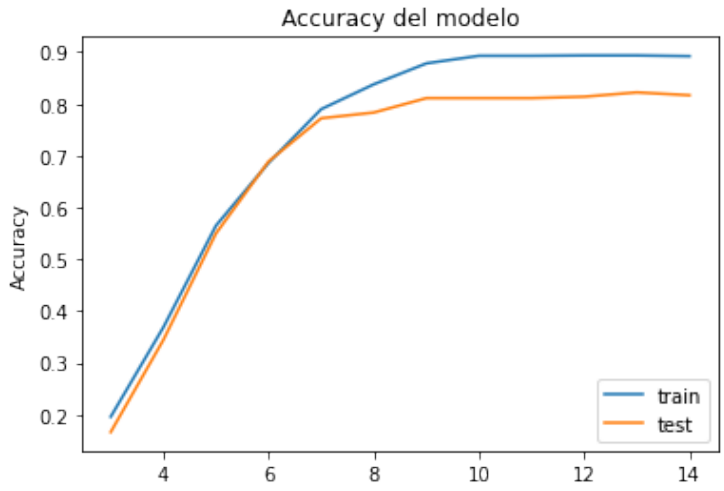
2.2.2 Cambiando gini a entropía



Ambos casos sobreentrenan, sin pruning el accuracy es de $\sim$ 85%. Con pruning, el accuracy para el test case disminuye entre gini y entropía.

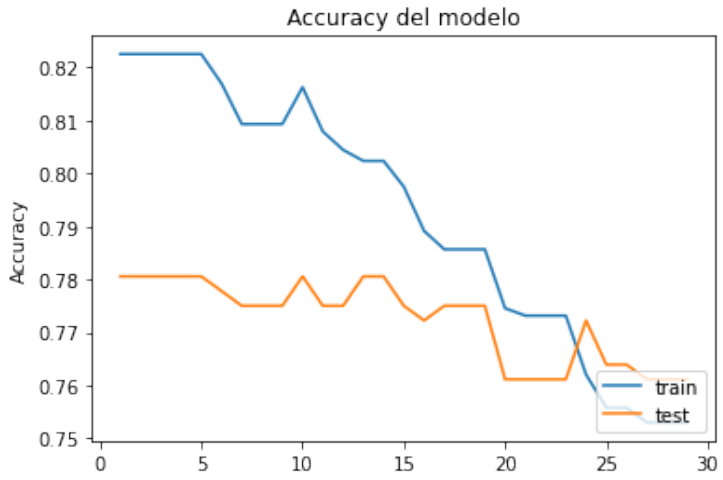
Se elige "entropy", los resultados no son significativamente diferentes.

2.2.3 Obteniendo la mejor profundidad máxima



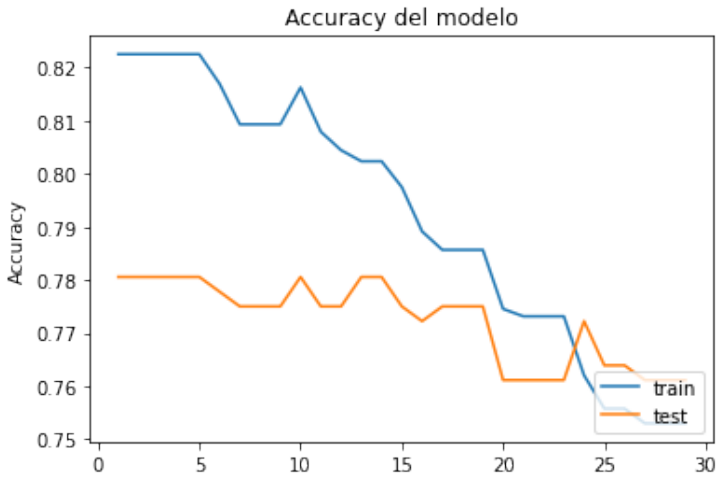
Se elige 8.

2.2.4 Obteniendo la mejor cantidad mínima de datos para un split



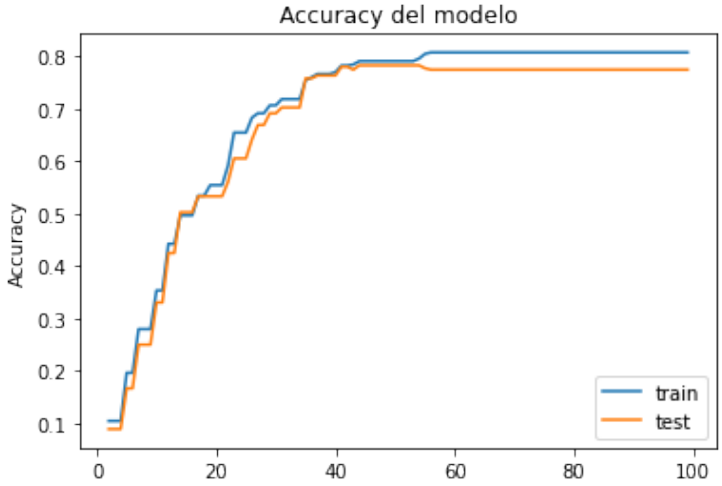
Se elige 15.

2.2.5 Obteniendo la mejor cantidad mínima de datos por hoja



Se elige 11.

2.2.6 Obteniendo la mejor cantidad máxima de hojas por nodo



Se elige 60.

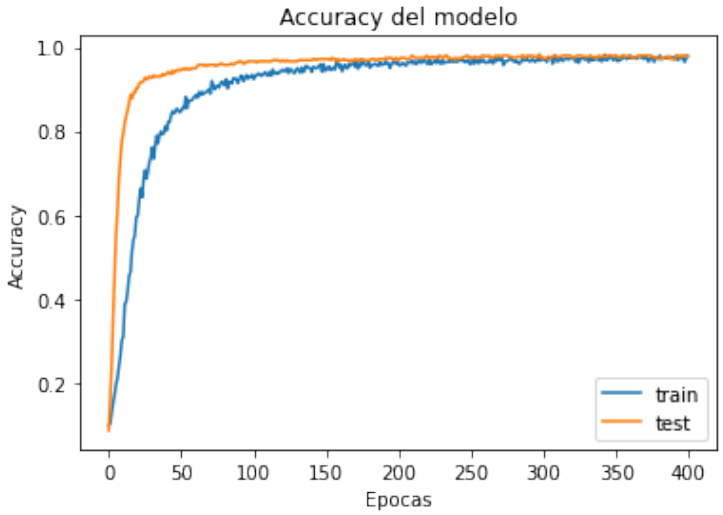
2.2.7 Resultados

Los resultados finales son 79.7% accuracy para training set y 77.5% para test set. Empíricamente, no es posible entrenar un árbol con precisión real (no generada por sobreentrenamiento) de mas del 80%, a partir de los resultados obtenidos

de los testeos. Esto se debe a que el árbol solo puede hacer divisiones lineales del espacio de posibles entradas, en este caso imágenes de números, esto limita fuertemente la capacidad de inferencia.

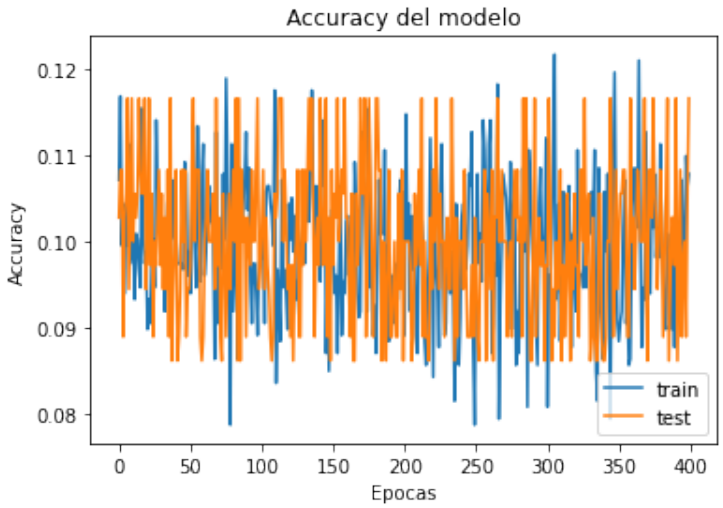
2.3 Redes neuronales

2.3.1 Modelo inicial, por defecto

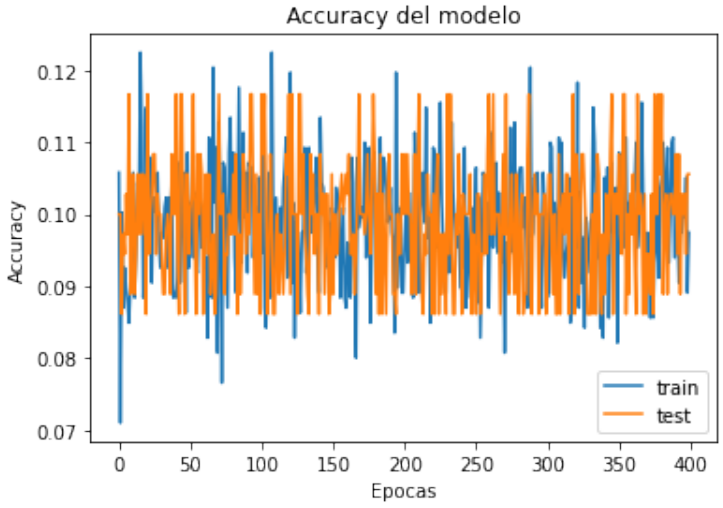


La red toma un vector de 64 valores de entrada, tiene una capas convolucionales con kernel $(3, 3)$ con 10 filtros, un capa Dropout para lidiar con el overfitting y luego tiene una salida softmax de ancho 10.

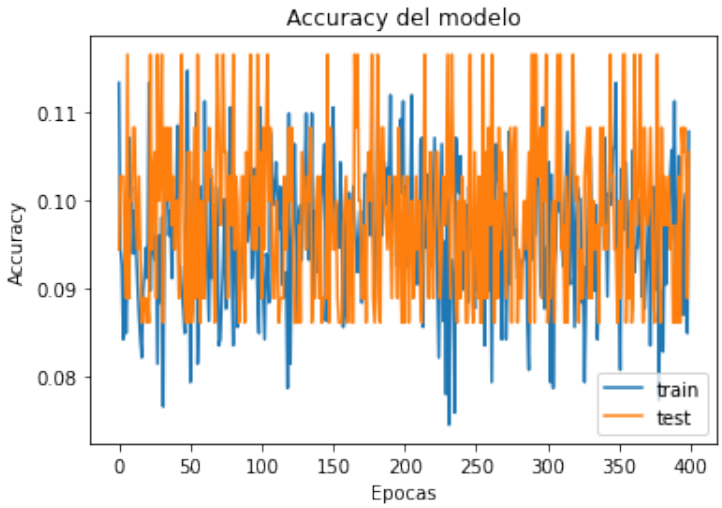
2.3.2 Learning rate 10



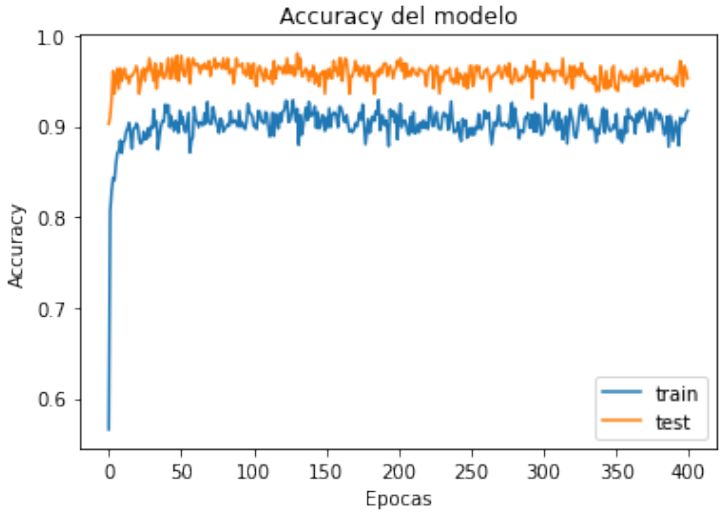
2.3.3 Learning rate 1



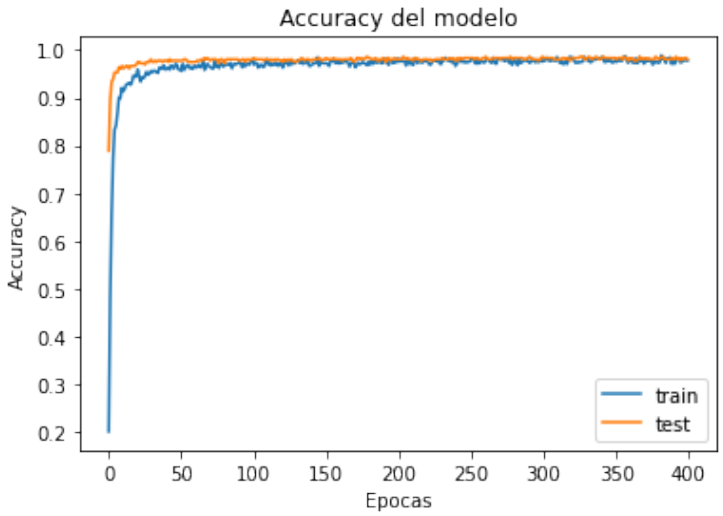
2.3.4 Learning rate 0.1



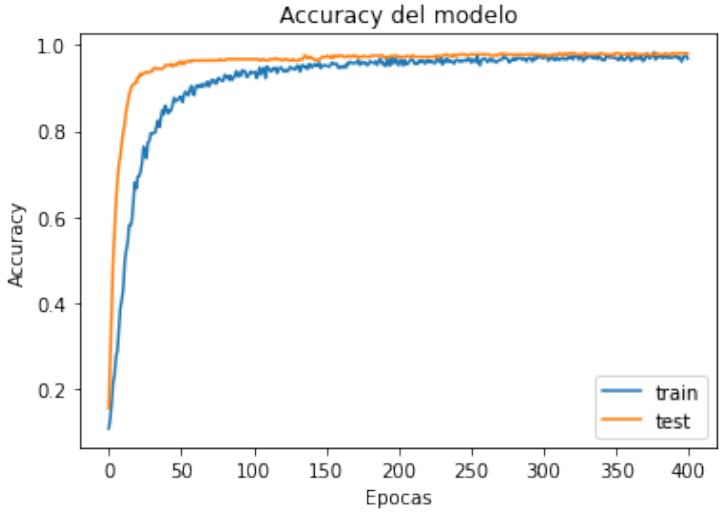
2.3.5 Learning rate 0.01



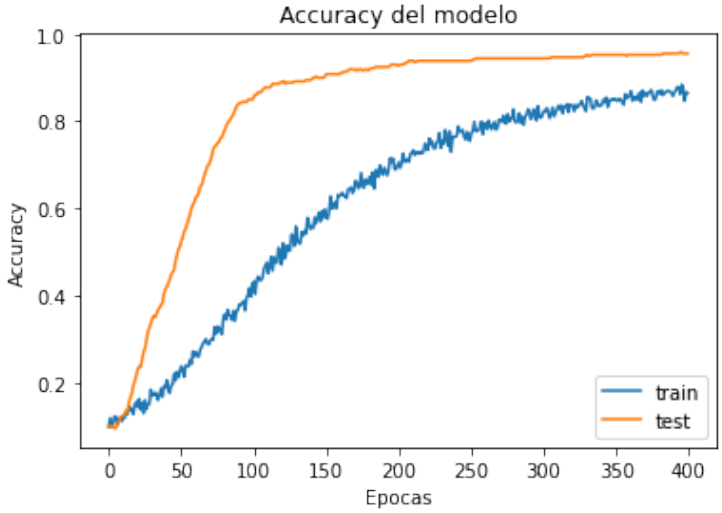
2.3.6 Learning rate 0.001



2.3.7 Learning rate 0.0001



2.3.8 Learning rate 0.00001



2.3.9 Resultados

Podemos apreciar como entre 0.01 y 0.0001 el modelo esta por encima del 90% de predicción correcta sobre el dataset de test en un tiempo relativamente rápido, pero notablemente peor sobre el dataset de entrenamiento, lo que nos dice que no se esta produciendo un sobreentrenamieto. Por encima del 0.01 el entrenamiento se vuelve errático, ya que no se puede bajar efectivamente por el gradiente, el paso que se da es siempre muy grande; de la misma forma con 0.00001 se tarda mucho mas en llegar a una accuracy aceptable ya que el descenso es muy lento (cada paso dado es muy chico).

3 Conclusiones

En este trabajo se puede ver claramente como el árbol de decisión esta muy limitado por sus características, que implican que solo puede hacer divisiones lineales del espacio de entradas, y, además no posee información sobre la cercanía de los elementos de entrada (los píxeles). Se vio como cuando el árbol aprende, este simplemente hace referencia a las características de cada píxel, y sobre entrena para clasificar solo las entradas de entrenamiento, es decir, no posee conocimiento sobre un "8" sino que hace una especie de caracterización de los "8" presentes en el dataset, esto produce que cuando se hace pruning la accuracy de testeo también baje, y que la capacidad de reconocimiento no generaliza.

En cambio, la red convolucional, con activaciones relu y la capa dropout, posee la capacidad de rápidamente aprender a distinguir números por las características locales de cada uno de los dígitos (los trazos, si se quiere), lo que permite una generalizabilidad "innata" y evita en todo momento el sobreentrenamiento.