TP Inicial - Laboratorio de Construcción de Software - Entrega 3

Ebertz Ximena, Gross Pablo, López Gonzalo

Version 1, 2023-08-27

Secciones

1. Introducción

En este documento, abordaremos las distintas dificultades encontradas en el entrenamiento del modelo de inteligencia artificial seleccionado, para detectar el nivel de retinopatía diabética a partir de imágenes de fondo de ojo. Explicaremos por qué este proyecto está fuera de nuestro alcance, y brindaremos una segunda alternativa para nuestro objetivo de desarrollar un modelo de inteligencia artificial con fines médicos. Detallaremos, también, el procedimiento realizado para llevar a cabo el desarrollo de esta alternativa.

2. Detección de nivel de Retinopatía Diabética a través de imágenes de fondo de ojo

Principalmente, nuestro objetivo fue analizar imágenes de fondo de ojo. Trabajamos con un dataset con gran cantidad de muestras, a alta resolución. Estas imágenes estaban etiquetadas mediante un archivo csv denominado `trainLabels_cropped`, que contenía el nombre de la imagen y su nivel de retinopatía diabética. Para trabajar con esta información, como primer instancia, limpiamos y le dimos el formato necesario a nuestros datos.

Para esto, como primera instancia calculamos la cantidad de clases y su tamaño, segun la información del archivo `trainLabels_cropped`. Como se puede ver teníamos cinco clases: 0, 1, 2, 3 y 4. Los datos estaban desbalanceados, lo que en el entrenamiento de nuestra IA generaría resultados desfavorables.

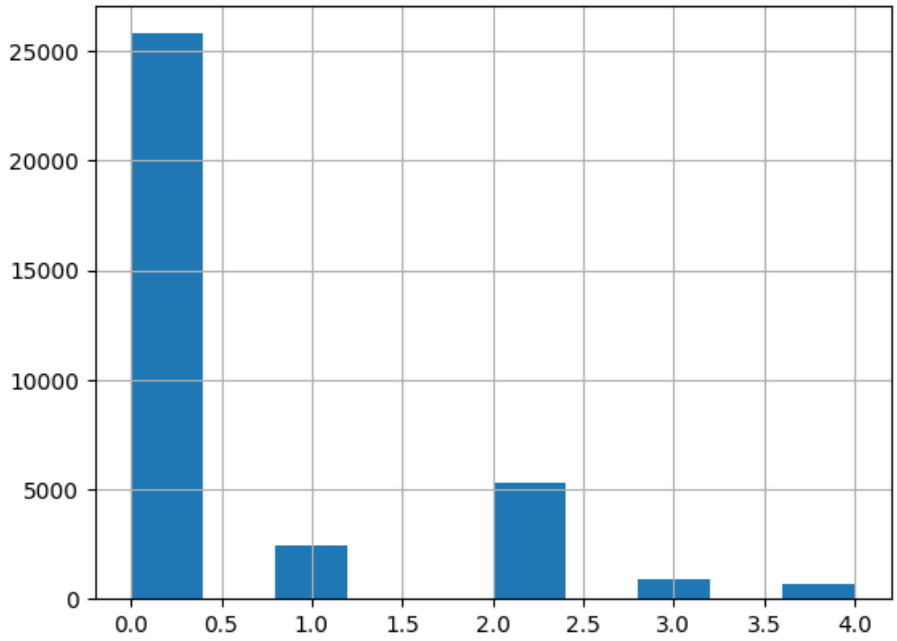

Figure 1. tamaño de clases

Como la clase 0 tiene muchos elementos, decidimos recortarlos, y dejar un aproximado de 4000 imágenes. Esto hizo que la distribución de las imágenes sea mas pareja.

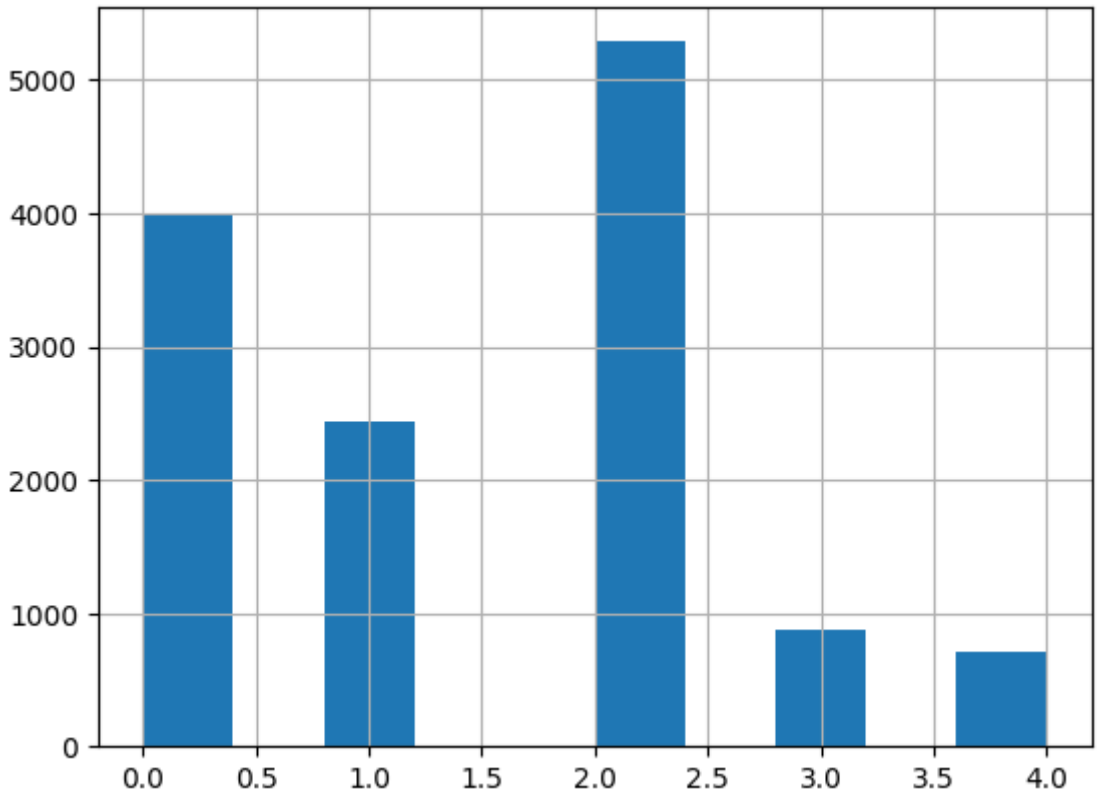

Figure 2. tamaño de clases con menos elementos de tipo 0

En este caso, las clases siguen estando desbalanceadas, pero como las clases 3 y 4 tienen muy pocos elementos, decidimos probar con esta distribución para ver si funcionaba.

Luego leimos las imágenes, guardándolas a todas en una lista. Acá es donde surgió el primer problema: la lectura para imágenes de más de 100x100 pixeles no se podía hacer, ya que se llenaba la memoria y se cortaba la ejecución del programa. Las imágenes de 100x100 pixeles tardaban demasiado en cargar, lo que hacía que el programa sea inutilizable. Por este motivo, decidimos que la resolución de las imágenes sea de 50x50 pixeles.

Dividimos a las imágenes y sus etiquetas en tres conjuntos, uno de entrenamiento, otro de validación, y otro de prueba. El conjunto de entrenamiento se utiliza para que la IA aprenda y reconozca patrones. El de validación se utiliza para que, en su entrenamiento, la IA analice esas imágenes, con la finalidad de calcular su nivel de error. El conjunto de prueba se utiliza para, luego del entrenamiento, utilizar a la IA para que prediga los resultados de estas imágenes, y verificar su funcionamiento. El conjunto de entrenamiento corresponde a un 65% de la totalidad de las imágenes, el de validación corresponde a un 15%, y el de prueba corresponde al 20% restante. Cada imagen fue normalizada, es decir, los valores de todos los pixeles fueron establecidos en el rango [0, 1].

Luego, las imágenes y etiquetas fueron convertidas a un formato reconocible por la IA, llamado *tensor*.

Cuando visualizamos las imágenes, nos dimos cuenta que la resolución de las imágenes era demasiado baja para que los humanos podamos detectar alguna anomalía en ellas. Sin embargo, procedimos igualmente a analizarlas con un modelo convolucional y uno denso.

El entrenamiento de ambos modelos fue demasiado lento, y con niveles de exactitud muy bajos, y niveles de error muy altos. Esto sugería que la IA no aprendía.

AGREGAR GRÁFICOS DE MÉTRICAS

Agregar o eliminar capas de neuronas, balancear mejor las clases o configurar los hipeparámetros de ambos modelos no sirvió, por lo que probamos con una técnica llamada *aumento de datos*. Esta técnica consiste en "deformar" las imágenes, cambiando su iluminación, zoom, rotación, etcétera. Se utiliza para mejorar el entrenamiento de los modelos de IA de análisis de imágenes, ya que aumenta el número de imágenes de entrenamiento y su variedad. Sin embargo, esto tampoco funcionó.

Ambos modelos tenían los mismos resultados, lo que no debería suceder ya que son

modelos distintos.

Luego de analizar y probar diversas técnicas, determinamos que el problema elegido escapa nuestro alcance, ya que las imágenes deberían analizarse a mayor resolución, por lo que se necesitan mas recursos, que no tenemos.

Como el objetivo del trabajo es crear un modelo de IA con fines médicos, decidimos cambiar el enfoque del proyecto. Nuestro nuevo objetivo es detectar si un lunar es benigno o maligno a partir de una imagen. Debido a que seguimos utilizando imagenes, podemos reutilizar todo el trabajo ya desarrollado, cambiando la fuente de datos. Esto hace que el tiempo final de desarrollo se reduzca considerablemente.

3. Detección de lunares benignos y malignos

El nuevo propósito de nuestro sistema radica en la detección precisa de lunares benignos y malignos a través del análisis de imágenes. Estas imágenes son adquiridas de la página web Kaggle, concretamente del conjunto de datos disponible en Skin Cancer: Malignant vs. Benign. Este conjunto específico consta de un total de 2637 imágenes utilizadas para el entrenamiento, distribuidas en 1440 imágenes de lunares benignos y 1197 imágenes de lunares malignos. Además, se dispone de 660 imágenes para llevar a cabo pruebas, compuestas por 360 imágenes de lunares benignos y 300 imágenes de lunares malignos.

3.1. Pre-Entrenamiento

Antes de llevar a cabo el entrenamiento del modelo, fue necesario ejecutar una serie de pasos para asegurar su viabilidad y efectividad.

Inicialmente, procedimos a descargar todas las imágenes disponibles desde la página web previamente mencionada.

Posteriormente, organizamos estas imágenes en listas separadas, categorizándolas en función de si serían destinadas para el entrenamiento o la fase de pruebas. Además, aplicamos una estandarización en las dimensiones, ajustando cada imagen a un formato de 100x100. Este enfoque se eligió para evitar consumir excesiva memoria RAM en el entorno de Google Colab.

Luego, cada imagen fue etiquetada en consecuencia. Aquellas que representaban lunares benignos se etiquetaron con un valor de 0, mientras que las imágenes de carácter maligno se etiquetaron con un valor de 1.

Con el propósito de evitar sesgos en el modelo, implementamos una etapa de mezcla de las imágenes. Esta mezcla se llevó a cabo de manera que las etiquetas continuaran alineadas correctamente. De esta manera, se evitó que el modelo recibiera secuencias de imágenes en las que las muestras benignas o malignas estuvieran agrupadas en bloques.

Además, llevamos a cabo una etapa de normalización en las imágenes. Esta normalización ajustó los valores de los píxeles en un rango entre 0 y 1, lo que resulta fundamental para un procesamiento y entrenamiento más eficiente del modelo.

Una vez completados estos pasos, estuvimos en condiciones de comenzar con el proceso de entrenamiento y llevar a cabo pruebas para evaluar el rendimiento del modelo resultante.

3.2. Entrenamiento de modelos

Realizamos el entrenamiento de redes neuronales densas y convolucionales, y a continuación, compartiremos las configuraciones de parámetros que empleamos para estas distintas redes, así como aquella que determinamos como el modelo óptimo.

3.2.1. Configuración de parámetros

Red neuronal densa

Esta red neuronal densa está configurada de tal manera que se tiene una capa de entrada de 10,000 neuronas, correspondiendo cada una de estas a un píxel de la imagen de 100x100 píxeles.

Después se tienen dos capas ocultas que contienen 150 neuronas cada una, las cuales se encargan de analizar los datos de las neuronas de entrada.

Por último, se encuentra una sola neurona de salida, la cual determina con un 1 o un 0 (redondeando los resultados intermedios) si el lunar de la imagen analizada es maligno o benigno.

Por ende, este modelo solo llega a alcanzar como máximo un 76/77% de precisión. Lo cual implicaría un alto nivel de precisión, pero por cómo actúa la red neuronal densa es que pierde bastante el contexto de las imágenes dadas. Por lo tanto, al procesar información que se encuentra fuera de los rasgos de las imágenes de entrenamiento, pierde eficacia y precisión.

Red neuronal convolucional

Esta red neuronal convolucional está configurada de tal manera que se tienen tres capas convolucionales en la entrada. Las cuales se encargan de observar la imagen en clústers de 3x3 píxeles, 3 veces, para poder comprimirlas manteniendo las características más importantes de la misma. Para así poder procesarla más rápido en las capas subsiguientes.

Después se encuentra la capa de dropout, la cual modifica los resultados de los nodos a los cuales se dirigen los resultados, para evitar sobrecompensación en los resultados.

Siguiendo a esto, se genera la capa de entrada, que toma la imagen comprimida y genera una neurona de entrada por cada píxel de la misma imagen, para así poder procesarla.

A continuación, se encuentra una capa oculta de 25 neuronas para poder procesar los datos de las imágenes.

Por último, se encuentra la capa de salida, la cual es una sola neurona que determina con un 1 o un 0 si el lunar es maligno o benigno.

Dadas las características de las capas convolucionales, se puede intuir que es recomendable usarlas para el análisis de imágenes, ya que permiten añadir contexto espacial a la predicción del modelo neuronal. Por esto mismo, el modelo que estamos usando llega a una precisión del 80/81%.

3.2.2. Modelo óptimo

Por lo mencionado previamente en la explicación de los modelos usados, se puede llegar finalmente a la conclusión de que para la tarea a completar, la cual consiste en analizar fotos, es más óptima la red neuronal convolucional. Esto se debe a que presenta un nivel mayor de precisión y permite que con el entrenamiento presentado para el modelo pueda intuir y determinar un resultado de una imagen con la cual no entrenó y que no sea completamente similar a un dato de entrenamiento.

Entrando en más detalle, la red neuronal densa en su aprendizaje puede llegar a un 78% de precisión, pero este resultado no se presenta en el testeo con datos aleatorios de los cuales no aprendió, lo que genera una variación grande en los resultados de sus predicciones.

Por otra parte, la red neuronal convolucional quizá tarde más en su entrenamiento, pero llega a un porcentaje de precisión del 81%, el cual también se traslada a

ejemplos del mundo real con datos aleatorios que no se encontraban en los datos de entrenamiento. A su vez, por el tipo de aprendizaje de contexto en las imágenes, permite una mayor consistencia en sus resultados, el cual también es 81%.

4. Conclusión

...