RT-BENE: detección de parpadeo

Esteve Soria Fabián

Febrero 2022

1 Resumen

Se generan dos modelos para la resolución del problema de detección de parpadeo.
El primero es un modelo ingenuo en el cual no tengo en cuenta la distribución de datos. Se selecciona un modelo de los disponibles en el zoo de Keras por simplicidad. Además el modelo está preentrenado en Imagenet.
El segundo modelo está también basado en un modelo preentrenado de Keras pero en este caso sí que se tiene en cuenta la distribución de las clases en el dataset y se utilizan técnicas de aumento de datos para corregir la sobre-representación de la clase "no parpadeo" con respecto a "parpadeo".
En este proyecto se consigue un F1-score de 0.99 en el testset.

2 Introducción

Se utiliza el conjunto de datos RT-BENE, este consiste en 107350 parejas de imágenes y su correspondiente etiqueta.
Cada pareja de imágenes contiene dos imágenes RGB de tamaño 36x60 pixeles de los ojos de diferentes personas. En total, el dataset, contiene 17 videos cada uno de una persona diferente donde se ha anotado en cada imagen si está parpadeando o no.
El problema a solucionar es la predicción por cada par de imágenes si estas corresponden a un parpadeo o no. Dicho problema se puede considerar como un problema de clasificación donde hay que predecir si para dicha entrada se corresponde una clase u otra. En este caso como solo hay una salida se define como clasificación binaria.

3 Métodos

Se proponen dos soluciones para dicho problema de clasificación. La primera aproximación es la más simple de aplicar y además es ingenua en el sentido de que no considera cual es la distribución de las clases en el dataset. En la segunda se utiliza un modelo más eficiente y además tiene en cuenta la distribución de las clases para poder hacer un entrenamiento más equilibrado.

Lectura de datos

El primer paso para resolver un problema de análisis de datos es su lectura misma ya que cada dataset viene en un formato diferente y por tanto no se puede seguir la misma aproximación en cada problema. En este dataset se tiene una carpeta con las imágenes y un fichero csv donde cada fila tiene los siguientes datos:

- Identificador único
- Ruta relativa a la imagen del ojo izquierdo
- Ruta relativa a la imagen del ojo derecho
- Identificador del video
- Clase a predecir. '1' si es parpadeo, '0' si no lo es.

Una vez se ha identificado que contiene el dataset es interesante ver la distribución de este según las variables disponibles. Se muestra cuantas imágenes hay en cada uno de los videos. Por ejemplo, al tener un video de cada persona no tenemos la misma cantidad de imágenes por cada persona. Si el desbalance es muy importante se podría sobreentrenar sobre una persona determinada y no ser posible predecir correctamente sobre una población ya que pertenece a un dominio diferente.
En el caso de este dataset no existe un video con una cantidad relativamente grande de imágenes con respecto a las demás por tanto no se rebalancea con respecto al identificador del video.
Otra observación del dataset que si que se tiene en cuenta es el balance de la clase a predecir. Se puede ver como en cada video solo hay entre un 1% y 10% de imágenes que corresponde a la clase parpadeo. Esto significa que si no se utilizan medidas para corregir el desbalance producirá un efecto negativo en la capacidad del modelo de predecir correctamente.

Partición de datos

Se parte el dataset en tres particiones. Una partición de entrenamiento, otra de validación y otra de test.
Primero se parte el dataset en dos partes, entrenamiento y test, dejando el 20% del dataset para test. Un valor de 20% es relativamente generoso para el problema donde normalmente se considera un valor entre 10% y 20%.

Por ejemplo una competición como Imagenet tienen un 15% del dataset como test[2]. La partición se hace mediante selección aleatoria ya que se considera que la selección estratificada no es necesaria ya que el dataset tiene una distribución de clases en cada video relativamente similar. Esto deja alrededor de 20.000 imágenes para test.
En la partición de entrenamiento se parte el dataset en dos para generar el set de validación. Se podría utilizar un sistema de k-fold para la validación pero en el entrenamiento de redes neuronales profundas se vuelve demasiado pesado para ser eficiente en el tiempo. El set de validación tiene un tamaño de 17.000 imágenes aproximadamente.

video	Images in video	% blink frames
0	12865	7.236689
1	8671	1.476185
2	8702	9.066881
3	3205	5.210608
4	4750	2.736842
5	5355	2.054155
7	1857	8.023694
8	6108	7.514735
9	4210	1.068884
10	16559	2.131771
11	12817	5.399079
12	935	2.459893
13	9586	3.077405
14	5371	4.002979
15	1810	1.602210
16	4549	0.923280

Figure 1: Tabla con imágenes por video y porcentaje de parpadeos por video

Lectura y generador de datos

Se crea una clase que permite cargar los datos del fichero .csv y permitir la carga dinámica de las imágenes antes de que sean necesitadas. Eso se hace mediante el uso de la librería tf.data .Dataset[1]. El generador de datos sigue el siguiente proceso:

```
print("La vida")
```

1.

4 Resultados

5 Conclusiones

References

[1] Derek G. Murray, Jiří Šimša, Ana Klimovic, and Ihor Indyk. Tf.data: A machine learning data processing framework. *Proc. VLDB Endow.*, 14(12):2945–2958, jul 2021.

[2] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 09 2014.