Detección de objetos utilizando una red neuronal R-CNN

Carla Viñas Templado, Hamza Akiour

Abstract— Sistema de detección de objetos utilizando una red R-CNN.

Keywords— PSIV, Kernel, Correlacion, detección de imágenes, objetos, mAP, CNN, R-CNN MatLab.

1 Introduccion

En el ámbito industrial, se utilizan frecuentemente programas capaces de detectar y contabilizar determinados objetos, de forma que se puede establecer un sistema de verificación. Nuestra motivación para este proyecto es ser capaces de desarrollar un sistema capaz de ser entrenado para detectar objetos con una precisión alta.

2 Estado del arte

2.1 Técnica actuales

Se ha realizado un previo trabajo de investigación para recolectar algunas de las principales técnicas actuales utilizadas para solventar està classe de problema, hemos agrupado estás en categorías: las técnicas clásicas, y las técnicas modernas.

2.2 Técnicas clasicas

Las técnicas clasicas pueden ser parámetricas, las cuales se utilizan cuando la función densidad de probabilidad se asemeja a alguna distribución conocida (Gaussiana, de Poisson, etc), o no parámetricos, es decir, independientes de la forma que toma la función densidad de probabilidad (a partir de ahora pdf).

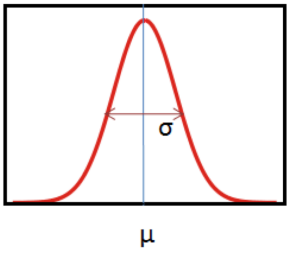

Fig. 1. Funcion Gaussiana definida mediante la media μ y la variancia σ^2.

Principalmente se basan en establecer el fondo de la imagen como modelo principal, una vez se substrae del modelo la imagen que está siendo evaluada, nos da las regiones que han sufrido una variación frente al modelo principal.

2.2 Técnicas modernas

Las técnicas modernas consisten en utilizar redes neuronales convolucionales las cuales detectan una cantidad limitada (ó éspecifica) de objetos dentro de nuestro contexto (las imágenes en nuestro caso). Hay varios metodos, nosotros trataremos principalmente los detectores de clasificación (como la que se utiliza en Alexnet). Aquí se presentan distintos problemas, ¿De qué tamaño será la ventana?, ¿qué desplazamiento lateral realizaremos?, ¿Cómo evitamos detectar el mismo objeto varias veces debido al desplazamiento? Debido a estás preguntas, nos ha resultado más interesante realizar el sistema de detección utilizando está familia de técnicas, en lugar de las técnicas clásicas.

3 Propuesta

De las preguntas planteadas en la sección 2.2, surge la necesidad de utilizar una métrica específica para la detección de imágenes en donde podamos evaluar si el objeto es correcto, y la posición de este. En nuestro caso utilizaremos la métrica mAP, definida como:

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

Antes de empezar a desarrollar el programa, hay que establecer la fuente de datos, el criterio de evaluación, entre otros (ver apartado 4).

Una vez sabemos las métricas a seguir, tenemos que establecer la estrategia que nos permita establecer estos (R-CNN, SSD, YOLO, …), en nuestro caso, sabemos que YOLO nos da un resultado mucho más veloz frente a R-CNN, pero como queremos hacer una solución industrial, nos importa más la precisión, que la velocidad, por ello desarollaremos este proyecto utilizando R-CNN.

Es decir, pasaremos un primer algoritmo sobre la imagen a tratar para determinar las áreas de interés, las cuales pasaremos por la CNN y mediante un clasificador (binario) validaremos las clases (para descartar las que no nos transmiten suficiente veracidad). Una vez en este punto, utilizaremos un regresor para terminar de ajustar la posición correcta.

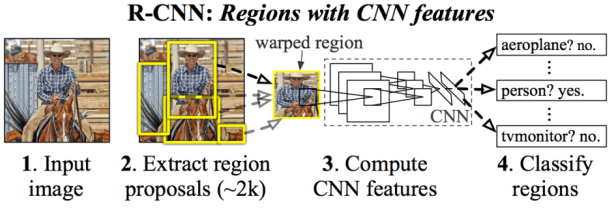

Fig. 2. Descripción visual del proceso de detección de clases mediante R-CNN.

Después aplicaremos IoU, el cual nos da un porcentaje de acierto del área de predicción frente al área real que queremos detectar. Finalmente, con NMS podremos detectar entre varias cajas que

colisionan entre si, la que mejor se ajusta al objeto que estamos detectando.

Una vez en este punto, es necesario comprobar que el resultado se adecua al buscado, para ello, se evaluaran manualmente un paquete de imágenes clasificadas, y se evaluará otro de forma automática el cual se contrastará con la información ya definida en el dataset que utilizamos, gracias a los metadatos que vienen anotados en las imágenes las cuales nos describen el objeto.

4 Experimentos, resultados i anàlisis

Partiendo de la base de que desconociamos las librerias de referencia para realitzar el desarrollo de este programa, hemos tenido que realitzar una extensa investigación sobre todas las alternativas que se adecuan a las necesidades que este proyecto presenta.

Hemos decidido realitzar el programa en MatLab, siendo concientes de que en Python hay mucha más información, se ha considerado que es más apropiado para nuestra formación y disfrute personal realizarlo en MatLab, teniendo en cuenta que todos los retos y labs han sido realizados en està plataforma.

El proyecto se ha dividido en cuatro fases:
Fuente de datos
Entrenamiento de la red
Testing
Ejecución

4.1 Fuente de datos

Para la creación de la red hemos analizado distintos datasets que nos permitían entrenar la red.

Finalmente nos hemos centrado en dos dataset; ImageNet y CIFAR-10. Tras evaluar estos, decidimos utilizar CIFAR-10 ya que nos ofrece una versión adaptada a MatLab de forma que podemos leer las etiquetas asociadas a cada imagen de forma sencilla.

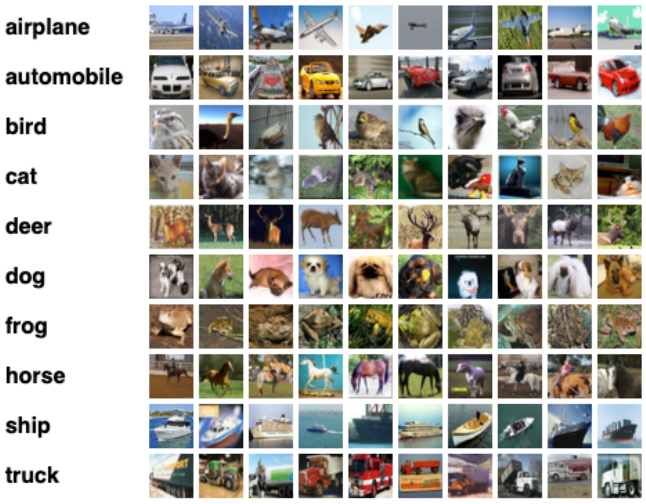

Fig. 3. Muestreo de 10 imágenes random de las 10 categorías

Posteriormente se ha descargado el dataset en formato tar.gz, este set está compuesto por 50 mil imágenes para el entrenamiento y 10 mil para el testeo.

Las imágenes tienen un tamaño de 32x32 pixeles y un peso medio de 2 KiloBytes.

Fig. 4. Imagen del dataset. Peso: 3 KB

Al obtener las fotos utilizando la versión adaptada de MatLab, estás continen las etiquetas de que label está asignado a cada foto, al leer los datos es importante no descartar estás etiquetas, ya que nos permitirán realizar el entrenamiento (un dataset sin un correcto etiquetamiento nos conducirá a un resultado pobre con muchos falsos positivos).

4.2 Entrenamiento de la red

Para el entrenamiento de la red R-CNN hemos creado distintas capas. La inicial continene el tamaño de las imágenes de muestra.

Después se ha incluído una capa convolucional con el tamaño del filtro de la imagen y un kernel (ventana) de 5x5, cabe recalcar que se ha puesto un padding de 2 para no perder información en los bordes.

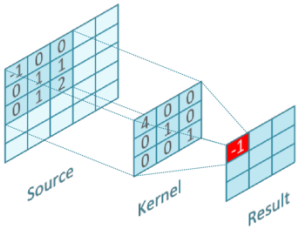

Fig. 5. Representación gráfica del kernel

Después se ha ñadido una capa relu, la cual nos permite filtrar nuestros datos, todos los valores inferiores a 0 obtienen el valor de 0, y los demás se quedan con el mismo valor.

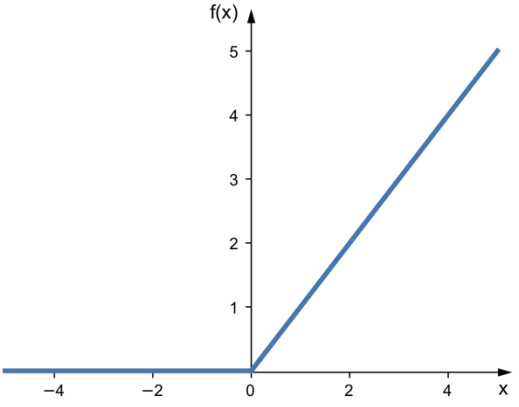

Fig. 6. Funcionamiento capa Relu. Transformación de datos.

A posterior añadimos una cuarta capa, la de polling, la cual nos permitirá quedarnos con las caracrteriscas más comunes. Repetimos este proceso (las 3 últimas capas) 3 veces. Durante las pruebas lo hemos hecho con 2 y con 4, con 2 el resultado no es tan preciso, con 4 la mejora no es tanta en comparación al tiempo que aumenta en la fase de entrenamiento.

Para las capas finales, añadimos una serie de capas que hará la función de activación, la cual nos permitirá realizar la clasificación.

A posterior inicializamos la convolución con pesos aleatorios utilizando una distribución normal.

En la configuración de la red se han establecido los siguientes parámetros de entrenamiento:

Etiqueta	Valor
Momentum	0.9000
InitialLearnRate	1.0000e-03
LearnRateSchedule	'piecewise'
LearnRateDropFactor	0.1000
LearnRateDropPeriod	8
L2Regularization	0.0040
GradientThresholdMethod	'l2norm'
GradientThreshold	Inf
MaxEpochs	40
MiniBatchSize	128
Verbose	1
VerboseFrequency	50
ValidationData	[]
ValidationFrequency	50
ValidationPatience	Inf
Shuffle	'once'
CheckpointPath	''
ExecutionEnvironment	'auto'
WorkerLoad	[+
OutputFcn	[]
Plots	'none'
SequenceLength	'longest''
SequencePaddingValue	0
SequencePaddingDirection	'right'
DispatchInBackground	0
ResetInputNormalization	1

Tabla 1. Parametros de configuración de la red neuronal

4.3 Testing

Para la fase de testeo lo que hemos hecho ha sido evaluar la red neuronal frente al set de imágenes de training. Después, para saber que precisión real tenemos, lo hemos comparado con el resultado que tendría que haber dado, obteniendo una precisión de 0.75, es decir, ¾ de las imágenes han sido identificadas con éxito.

4.4 Ejecución

Para la ejecución de nuestro programa, lo que se ha realizado ha sido utilizar un set de imágenes, en este caso 41 imágenes con señales de stop para entrenar el nuevo objeto a identificar en la red. El entrenamiento realizado es similar al anterior (punto 4.2), podeis ver más información al respecto en el GitHub de este proyecto.

Etiqueta	Valor
InitialLearnRate	1.0000e-03
LearnRateDropPeriod	100
L2Regularization	1.0000e-04
MaxEpochs	100

Tabla 2. Parametros modificados en las opciones de entrenamiento de la señal de stop

Una vez entrenado, hemos cogido una imagen de muestra en la que sabemos que hay una señal de stop y su localización, para evaluarla con nuestro programa. Como se puede apreciar en la figura 7, se ha identificado de forma exitosa la seña de stop.

Hemos realizado varias pruebas, como añadir ruido a la imagen, y el resultado es cuanto menos sorprendente (positivamente). La precisión es muy alta.

Fig. 7. Detección de señal de stop con una precisión de 0.996% utilizando nuestro programa.

CONCLUSIÓN

Este proyecto nos ha servido para entender como funciona una red neuronal, matizar los conceptos aprendidos en clase y palpar el gran cambio que las redes neuronales suponen y supondrán de cara a un futuro.

Ha sido un proyecto ambicioso, el cual ha sufrido varios cambios importantes a lo largo de su desarrollo, el área a explorar está aún muy verde en comparación a otras áreas en las cuales hay mucha información.

Hay que matizar que este proyecto no habría sido posible si no se tuviésemos el dataset de Cifar-10, una red neuronal cuando ya está entrenada o preparada para ser entrenada hace que cuando se le enseña un objeto, lo aprenda de forma más sencilla. Hemos hecho pruebas (como utilizar solo un objeto) y el resultado es muy pobre, tener un dataset preparado hace que la red neuronal sea mucho más precisa.

Los dos miembros del grupo nos hemos quedado con un buen sabor de boca, hemos disfrutado el proceso a la vez que aprendido mucho. No hay nada más grato que cumplir el objetivo que se había propuesto, llegar a identificar 'algo' utilizando una red neuronal ha sido muy gratificante.

BIBLIOGRAFIA

[1] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman https://homepages.inf.ed.ac.uk/ckiw/postscript/ijcv_voc09.pdf

[2] Cerebrou - What is the mAP metric and how is it calculated? https://stackoverflow.com/questions/36274638/what-is-the-map-metric-and-how-is-it-calculated

[3] Héctor López Paredes "DETECCIÓN Y SEGUIMIENTO DE OBJETOS CON CÁMARAS EN MOVIMIENTO" http://arantxa.ii.uam.es/~jms/pfcsteleco/lecturas/20110930HectorLopezParedes.pdf

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik https://arxiv.org/abs/1311.2524

[5] Carla Viñas Templado, Hamza Akiour – Código fuente del programa https://github.com/hakiour/Detecci-n-de-objetos-utilizando-una-red-neuronal-R-CNN