Text detection using Keras-RetinaNet

Descrizione del problema

Ci siamo occupati del riconoscimento di testo. Lo scopo è dunque quello di fornire in ingresso un’immagine contenente del testo e ricavare la presenza del testo; oppure se l’immagine non contiene il testo viene fornita l’informazione che non c’è testo.

Dataset utilizzati:

* MSRA Text Detection 500 (MSRA-TD500)
* ICDAR

CSV:

* Pixel centro(x\_centro,y\_centro)

**1. Dataset**

1.1 **MSRA Text Detection 500 Database (MSRA-TD500)**

MSRA Text Detection 500 Database (MSRA-TD500) è un dataset rilasciato pubblicamente per valutare algoritmi di text detection, con lo scopo di tener traccia dei recenti progressi nel campo del text detection con immagini naturali, in particolare gli ultimi sviluppi nella rilevazione del testo di orientazioni arbitrarie.

MSRA Text Detection 500 Database (MSRA-TD500) contiene 500 immagini naturali, che sono state ricavate da scene al chiuso (ufficio e supermercato) e all’aperto usando una fotocamera. Le immagini di scene al chiuso sono principalmente segnali stradali, targhe di porta e targhe di cautela/piastra di cautela/lastra di cautela, mentre le immagini all’aperto sono principalmente tabelle di guida e manifesti in contesti complessi. Le risoluzioni delle immagini possono variare da 1296x864 a 1920x1080.

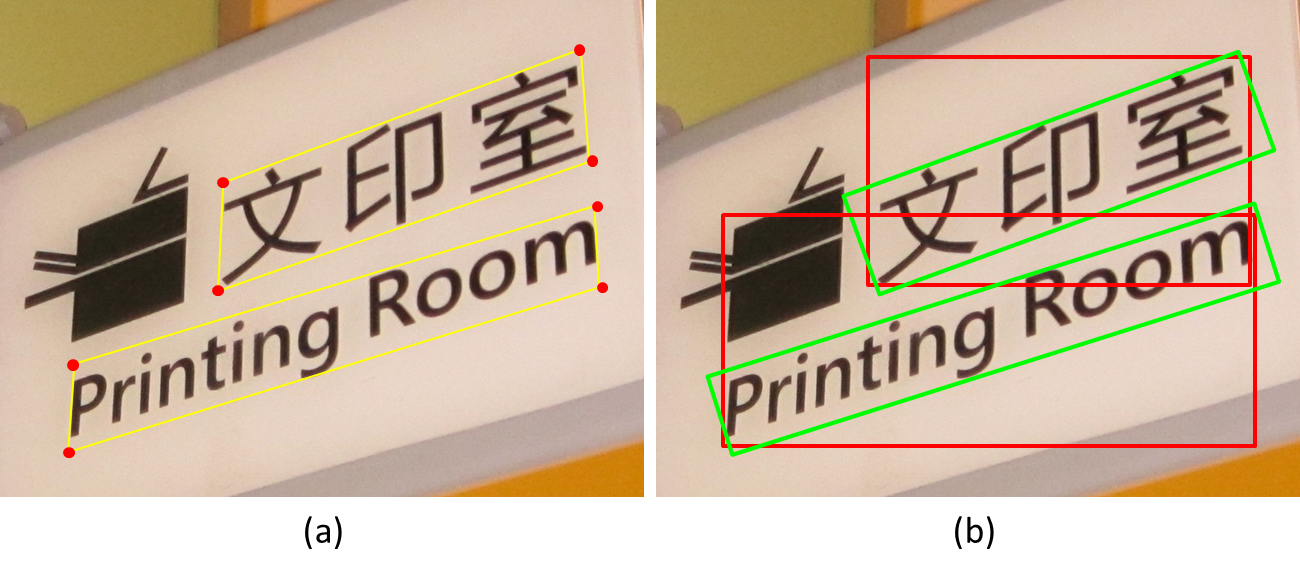
Il dataset MSRA è molto interessante sia per la diversità dei testi sia per la complessità dei contesti nelle immagini. I testi presenti sono di diverse lingue (Cinese, Inglese o misto dei due), font, dimensione, colori e orientazione. Il contesto può contenere vegetazione (per esempio alberi e prato) e pattern ripetuti (per esempio finestre e mattoni), che sono poco distinguibili dal testo.

Delle tipiche immagini con i corrispondenti rettangoli ground truth sono mostrati in figura.



**Figura 1. Tipiche immagini contenute nel dataset MSRA-TD500. Nota i rettangoli rossi. Questi indicano che il testo entro il rettangolo rosso è indicato come difficile (a causa di sfocature o occlusioni).**

Il dataset è diviso in due parti: training set e test set. Il Training set contiene 300 immagini selezionate in modo random dal dataset originale mentre il test set contiene 200 immagini. Tutte le immagini del dataset sono annotate. Nel dataset, l’unità base è la linea di testo (vedi Figura 1) piuttosto che singole parole, come invece è usato nei dataset ICDAR, perché è complicato dividere le linee di testo in lingua Cinese in singole parole basandosi sullo spazio tra di loro; anche per linee di testo in lingua Inglese, non è semplice dividere linee di testo in parole senza utilizzare informazione aggiuntiva.



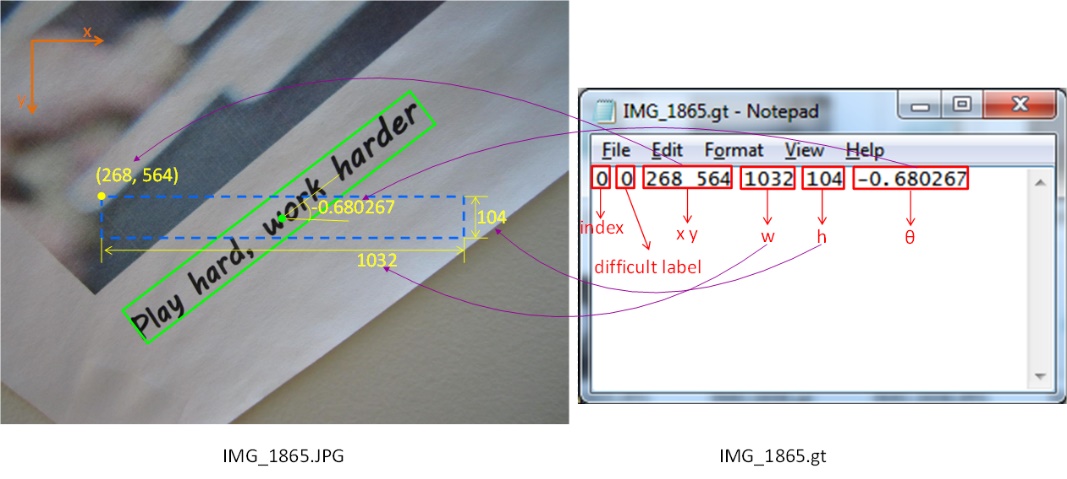
**Figura 2. Generazione ground truth. (a) Annotazione eseguita a mano. Le annotazioni sono richieste per localizzare e circoscrivere ogni linea di testo usando un poligono a 4 vertici (pallini rossi e linee gialle). (b) Rettangoli ground truth (verde). I rettangoli ground truth sono generati automaticamente riempiendo il rettangolo a area minima utilizzando il poligono.**

La procedure di generazione della ground truth è mostrata in figura 2. Mentre i metodi correnti di valutazione per la text detection sono progettati solo per testi orizzontali, noi proponiamo un nuovo protocollo di valutazione (vedi [1] per dettagli). Nel nostro protocollo vengono utilizzati rettangoli a area minima perché questi (i rettangoli verdi in figura 2 (b)) sono molto più contenuti rispetto a rettangoli allineati agli assi (i rettangoli rossi in Figura 2. (b)).

In particolare, per riconoscere testo particolare (o troppo piccolo, occluso, sfocato, troncato) che è difficile da riconoscere per algoritmi di text detection, abbiamo aggiunto a ogni testo considerato difficile da riconoscere un’etichetta aggiuntiva “difficult” (osserva il rettangolo rosso in figura 1.): il mancato riconoscimento di questi testi non viene considerato errore.

**Formato della ground truth**

A ogni immagine del dataset è associato un file ground truth, nel quale ogni record memorizza l’informazione di un testo. Il formato di un file ground truth è mostrato in figura 3.



**Figura 3. Illustrazione del file ground truth. Il campo index può essere ignorato. La difficult label è “1” se il testo è stato etichettato come “difficult”, altrimenti è “0”.**

Riferimenti

[1] C. Yao, X. Bai, W. Liu, Y. Ma and Z. Tu. Detecting Texts of Arbitrary Orientations in Natural Images. CVPR 2012.

**2. Creazione file CSV per il training**

2.1 **Architettura**

**3. Keras-Retinanet**

Link: <https://developers.arcgis.com/python/guide/how-retinanet-works/>

RetinaNet è una dei migliori modelli di riconoscimento one-stage che è stata testata per lavorare bene con oggetti densi e di piccola scala. Per questo motivo, RetinaNet è diventata un modello di riconoscimento oggetti diffuo per essere usato con immagini aeree e satellitari.

RetinaNet è stata definita sulla base di 2 miglioramenti rispetto agli altri modellidi riconoscimento oggetti a singolo stadio (single stage) – Feature Pyramid Networks (FPN)[1] e Focal Loss[2]. Prima di addentrarci nell’architettura, cerchiamo di capire prima cos’è Feature Pyramid Networks.

Per seguire la guida seguente, assumiamo che hai delle conoscenze di base dei concetti riguardanti le reti neurali convoluzionali (CNN). Puoi rivedere alcuni concetti riguardanti le CNN in questo breve paper “[A guide to convolution arithmetic for deep learning](https://arxiv.org/pdf/1603.07285.pdf)”. Si assume che il lettore abbia conoscenza dei modelli di riconoscimento oggetti. Fare riferimento alla guida “[How SSD Works](https://developers.arcgis.com/python/guide/how-ssd-works/)” dove i concetti come gli anchor boxes e feature maps sono spiegati.

3.1 **Cos’è una Feature Pyramid Network?**

Generalmente, nella computer vision, featurized image pyramids sono state usate per riconoscere oggetti con scale diverse dentro un’immagine. Featurized image pyramids sono feature pyramids costruite su image pyramids. Questo significa che si dovrebbe prendere un’immagine e sottocampionarla in immagini a via via minore risoluzione e dimensioni (formando quindi una piramide). Questa procedura rende la piramide invariante per scala. Di contro il processo è oneroso di risorse computazionali e di memoria.

Con l’avvento del deep learning, le features hand-crafted sono state rimpiazzate dalle CNN. Più tardi, la stessa pyramid feature è stata fatta derivare dalla stuttura gerarchica piramidale delle CNN. In una architettura di una rete neurale convoluzionale, la dimensione di uscita delle feature maps diminuisce dopo ogni blocco di operazioni convoluzionali, formando una struttura piramidale.

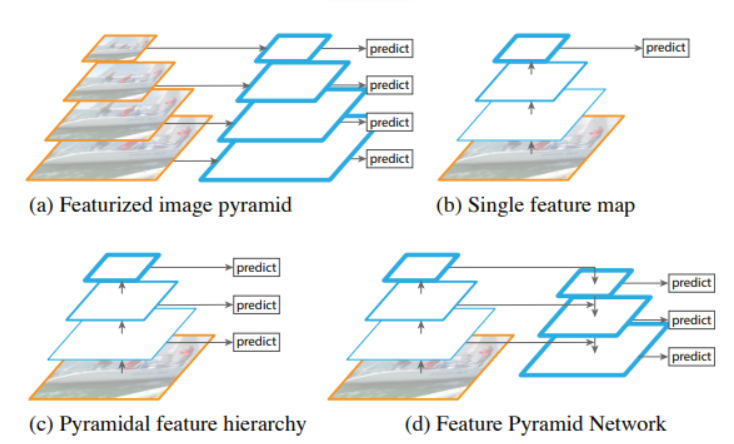


Figura 2. Different types of pyramid architectures [1]

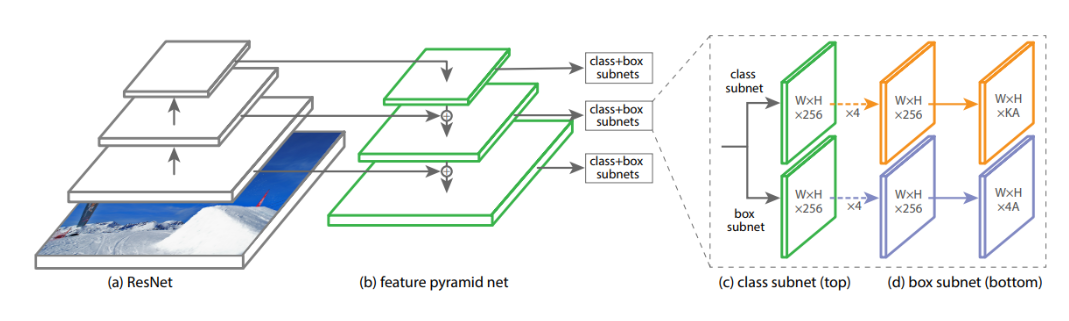
Ci sono diverse architetture CNN che utilizzano la struttura piramidale (Figura 2). La (a) mostra la Featurized image pyramid, che come abbiamo detto è onerosa di risorse. La (b) mostra una feature map singola (scalata), che è possibile usare per fare predizioni veloci. Nonostante siano più robuste e veloci, le featurized image pyramids sono utilizzate per ottenere risultati più accurati [1]. La figura (c) mostra che la Pyramidal feature hierarchy è stata usata da modelli come il Single Shot Detector, ma non usa le feature map multiscala di livelli diversi. La figura (d) è una Feature Pyramid Network(FPN) che compensa le carenze di queste variazioni. FPN definisce un’architettura con semantica ricca a tutti i livelli, in quanto combina feature semanticamente forti a bassa risoluzione con feature semanticamente deboli ad alta risoluzione [1]. Questo viene raggiunto creando un percorso top-down con connessioni laterali con livelli convoluzionali bottom-up.

Percorso top-down, percorso bottom-up e connessioni laterali saranno comprese meglio nella prossima sezione, dove andremo a vedere l’architettura RetinaNet. RetinaNet incorpora Feature Pyramid Network e aggiunge sottoreti per la classificazione e regressione per creare un modello per il riconoscimento oggetti.

3.2 **Architettura**

L’architettura di RetinaNet è composta da 4 componenti principali:

1. Percorso Bottom-up – la rete di backbone (per esempio ResNet) calcola le feature maps a scale differenti, a prescindere dalla dimensione dell’immagine in input o del backbone.
2. Persorso Top-down e connessioni laterali – il percorso top-down sovracampiona le feature map grezze dagli alti livelli della piramide, mentre le connessioni laterali fondono i livelli top-down e bottom-up con le stesse dimensioni spaziali.
3. Sottorete per la classificazione – predice la probabilità di un oggetto di essere presente in ogni locazione spaziale e per ogni anchor box e classe.
4. Sottorete per la regressione – per ogni oggetto della ground truth, ha il compito di regredire gli offset dei bounding boxes a partire dagli anchor boxes. (It's regresses the offset for the bounding boxes from the anchor boxes for each ground-truth object.)

****

3.2 **Focal Loss**

Focal Loss(FL) è la versione migliorata della Cross-Entropy Loss(CE) ed è stato introdotto per gestire il problema dello squilibrio presente dei modelli di object detection a Singolo stadio (Single Stage object detection models).

I modelli a singolo stadio soffrono del problema dello squilibrio di classi foregroung-background dovuto a eccessivo campionamento di anchor boxes (che potrebbero essere possibili locazioni oggetti). Con la rete RetinaNet, ci possono essere migliaia di anchor boxes in ogni livello piramidale. Solo pochi di questi anchor boxes verranno assegnati a oggetto della ground-truth e la maggior parte sarà la classe associata al background. Nonostante questi semplici esempi (predizione con alte probabilità) presentano valori di loss piccoli possono appesantire il modello. La Focal Loss riduce il contributo alla loss dovuti ai semplici esempi e aumenta l’importanza della correzione di esempi classificati erroneamente.

Riferimenti

* [1] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan: “Feature Pyramid Networks for Object Detection”, 2016; [[http://arxiv.org/abs/1612.03144 arXiv:1612.03144](http://arxiv.org/abs/1612.03144)].
* [2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He: “Focal Loss for Dense Object Detection”, 2017; [[http://arxiv.org/abs/1708.02002 arXiv:1708.02002](http://arxiv.org/abs/1708.02002)].

**4. Addestramento RetinaNet**

4.1 **Addestramento**

!keras\_retinanet/bin/train.py --freeze-backbone --random-transform --no-resize --weights {PRETRAINED\_MODEL} --batch-size 1 --steps 550 --epochs 40 csv annotation.csv classes.csv

**5. Caricamento del modello addestrato**

5.1 **Caricamento**

**6. Predizione vs ground truth**

6.1 **Predizione**