



Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Rilevamento di elementi testuali in immagini digitali

Relatore: *Prof. Raimondo Schettini*

Co-relatore: *Dott. Marco Buzzelli*

Relazione della prova finale di:

Manuel Rota

Matricola 793785

Anno Accademico 2016–2017

Sommario

Il rilevamento del testo in scene reali è un problema specifico della Computer Vision che negli anni è stato sempre oggetto di studi data la sua grande importanza, seppur mai completamente risolto al di fuori di applicazioni specifiche in condizioni controllate. In questa tesi analizziamo i diversi dataset a nostra disposizione per affrontare questo problema, introducendo brevemente anche le reti neurali con particolare attenzione alla loro utilizzo nell'ambito dell'object detection. Infine presentiamo vari dei nostri approcci sviluppati basati sulle *Fully Convolutional Networks* confrontandoci con metodi concorrenti ed ottenendo risultati discreti ma al di sotto degli standard odierni, seppur promettenti e che in futuro potranno certamente migliorare applicando le novità recenti susseguitesi nello stato dell'arte.

Indice

1	Introduzione	3
1.1	Competizioni	3
1.2	Stato dell'Arte	4
1.3	Struttura della tesi	5
2	Dataset e Competizioni	6
2.1	Dataset	7
2.1.1	Focused Scene Text	7
2.1.2	Incidental Scene Text	7
2.1.3	COCO Text	9
2.1.4	SynthText	10
2.2	Metodi di valutazione	12
2.2.1	Precision & Recall	12
2.2.2	Intersection over Union	13
3	Reti Neurali	14
3.1	Reti Neurali Artificiali	14
3.1.1	Perceptron	15
3.1.2	Backpropagation	16
3.1.3	Multilayer Perceptron	16
3.1.4	Funzioni di Loss	18
3.1.5	Apprendimento	19
3.1.6	Ottimizzazioni	20
3.2	Reti Neurali Convoluzionali	22
3.2.1	Struttura delle CNN	22
3.2.2	Breve storia	23
3.2.3	Casi di studio	24
3.3	Fully Convolutional Network	26
4	Eperimenti	28

4.1	Descrizione del metodo	28
4.1.1	Strumenti	29
4.1.2	Groundtruth	30
4.1.3	Iperparametri	30
4.1.4	Dettagli preliminari	31
4.2	Addestramento sui dataset	34
4.2.1	COCO-Text	34
4.2.2	Incidental Scene Text	36
4.2.3	SynthText	36
4.3	Modifiche strutturali	37
4.3.1	BranchFCN	38
4.3.2	Word Division	40
4.4	Finetuning e Data Augmentation	42
4.4.1	Finetuning	42
4.4.2	Data Augmentation	44
4.4.3	Confronto tra trasformazioni	45
4.4.4	Addestramento finale	46
4.4.5	Tabelle riassuntive	46
4.5	Confronto con lo stato dell'arte	49
4.5.1	Incidental Scene Text	50
4.5.2	Focused Scene Text	51
5	Conclusione e sviluppi futuri	52
5.1	Risultati	52
5.2	Sviluppi futuri	53

Capitolo 1

Introduzione

Il rilevamento del testo è stato per molti anni una sfida impegnativa nel campo della Computer Vision, soprattutto qualora questo non si trovasse inquadrato come elemento principale in un’immagine o fosse ubicato in contesti caotici riscontrabili nella vita quotidiana.

Infatti, per molti decenni le uniche implementazioni di tali metodi furono sempre rilegate a compiti molto ristretti e in condizioni prevedibili, come per esempio la localizzazione, estrazione e riconoscimento di parole in documenti stampati o scritti a mano, raggiungendo così risultati sì considerevoli, ma solo all’interno della specifica applicazione.

Scopo di questa tesi, come quello della ricerca in questo campo negli ultimi anni, è dunque il rafforzamento del task di **text detection** applicato a immagini raffiguranti i più vari contesti di scene reali andando a studiare ed applicare alcuni dei più recenti strumenti a nostra disposizione.

1.1 Competizioni

Nel corso degli anni, al fine di incentivare la ricerca in questo campo, sono nate molte competizioni riguardanti l’argomento in studio nelle sue più varie sfaccettature.

L’**International Conference on Document Analysis and Recognition** [1] (ICDAR), è una conferenza internazionale la quale dal 1991 a cadenza biennale raggruppa varie di queste competizioni.

Molti di queste sono strutturate perlopiù nei seguenti task individuali:

- Text Localization:
Ottenimento di un'area approssimativa occupata dalle zone di testo
- Test Segmentation:
Segmentazione dei singoli caratteri dallo sfondo
- Word Recognition:
Raggruppamento dei caratteri in parole e loro trascrizione

La fase di interesse di questa tesi è quella di *Text Localization*.

1.2 Stato dell'Arte

Negli anni molti metodi hanno tentato di approcciarsi a questo problema, adottando solitamente una fase di localizzazione dei singoli caratteri seguita da un loro raggruppamento in parole. Tra i più recenti possiamo citare:

- **TextFlow [2]**
Fast Cascade Boosting seguito da una rete di flusso dei costi minimi.
- **Canny Text Detector [3]**
Regioni estremali stabili massimamente (MSER) con AdaBoost per l'eliminazione di falsi positivi.

Recentemente però, con la crescita in popolarità e delle prestazioni delle reti neurali, ed in particolare quelle convoluzionali nel campo dell'object detection, sono nati molti altri metodi anche fra di loro differenti, fra i quali possiamo citare:

- **Multi-Oriented Text Detection [4]**
Region proposal con Convolutional Neural Networks seguita da metodi “tradizionali”
- **WordFence [5]**
Separazione delle zone proposte come testo con la predizione dei bordi delle parole
- **Deep Direct Regression [6]**
Proposta di zone di interesse e regressione delle singole bounding box

1.3 Struttura della tesi

Questa tesi è strutturata nei seguenti capitoli principali:

2. Dataset e Competizioni

Questo capitolo descrive i vari dataset pubblici utilizzati, sia per una nostra fase di addestramento che per la valutazione dei nostri metodi, analizzando le tecniche di misurazione di performance utilizzate.

3. Reti Neurali

Questo capitolo copre in breve la storia e teoria delle reti neurali dalla loro concezione fino ai giorni nostri, per arrivare al loro utilizzo odierno nel campo della Computer Vision per l'object detection.

4. Esperimenti

Questo capitolo illustra i nostri approcci incrementali al problema della Text Localization con le tecniche di *deep learning* illustrate nel capitolo precedente.

Capitolo 2

Dataset e Competizioni

Nonostante negli anni nel mondo della Computer Vision vi sia sempre stata una necessità di dataset pubblici per la comparazione di metodi concorrenti, i dati a disposizione su cui lavorare sono stati invece relativamente esigui poiché la gran parte dei problemi venivano solitamente risolti con un approccio di tipo analitico.

Da quando però nell'ultima decade è avvenuto un cambiamento nel modo di porsi a molte sfide grazie all'adozione delle reti neurali come strumento per l'apprendimento automatico, questa esigenza, aiutata anche dell'arrivo dei cosiddetti *Big Data*, si è espansa in svariati settori dell'informatica.

Il bisogno sempre più crescente di dati, nella nostra area di interesse di tipo visivo, però non è esente da problematiche: per l'apprendimento automatico sono infatti richiesti dati costituenti una *groundtruth*, ovvero informazioni che rappresentino verosimilmente la soluzione del problema postosi, e perciò non generabili in gran quantità ed automaticamente da un elaboratore.

Dataset degni di nota nel campo della Computer Vision che si sono imposti come standard nell'ambito dell'*object detection* sono il PASCAL Visual Object Classes [7], ImageNet [8] ed il più recente MSCOCO [9].

Di seguito approfondiremo i vari dataset a nostra disposizione per il task della localizzazione del testo e i principali metodi di valutazione proposti nella letteratura e adottati da varie competizioni.

2.1 Dataset

2.1.1 Focused Scene Text

Introdotto per la prima volta in occasione di ICDAR2013 [10], è composto da immagini acquisite in contesti reali dove l'oggetto principale è un'area di testo, solitamente orizzontale e formata da caratteri alfanumerici.

Nasce quindi avendo in mente i casi d'uso più comuni, quali la lettura del testo ed eventualmente una sua traduzione, portando dunque in secondo piano la fase di localizzazione, resa più immediata, a favore di un focus sulla precisa segmentazione e riconoscimento delle singole parole e/o lettere.

Caratteristiche:

- 229 immagini di training
- 233 immagini di test
- groundtruth composta dalle coordinate delle bounding box ortogonali e trascrizione di ogni singola parola

2.1.2 Incidental Scene Text

Introdotto per la prima volta nell'omonima competizione in occasione della conferenza ICDAR2015 [11], è composto da immagini acquisite in contesti reali (attraverso fotocamere indossabili) dove le aree di testo non sono l'oggetto principale e possono dunque trovarsi a diverse distanze e posizioni.

Questa differenza con il dataset precedente lo rende più adatto come metro di valutazione per il task su cui ci concentreremo in questa tesi in quanto la fase di localizzazione viene resa meno triviale. Verrà usato anche per la comparazione con metodi concorrenti.

Caratteristiche:

- 1000 immagini di training
- 500 immagini di test
- groundtruth composta dalle coordinate dei vertici delle bounding box orientate e trascrizione di ogni singola parola (quando leggibile)



(a) Focused Text Scene

(b) Incidental Text Scene.

Figura 2.1: Immagini di training con relative annotazioni.

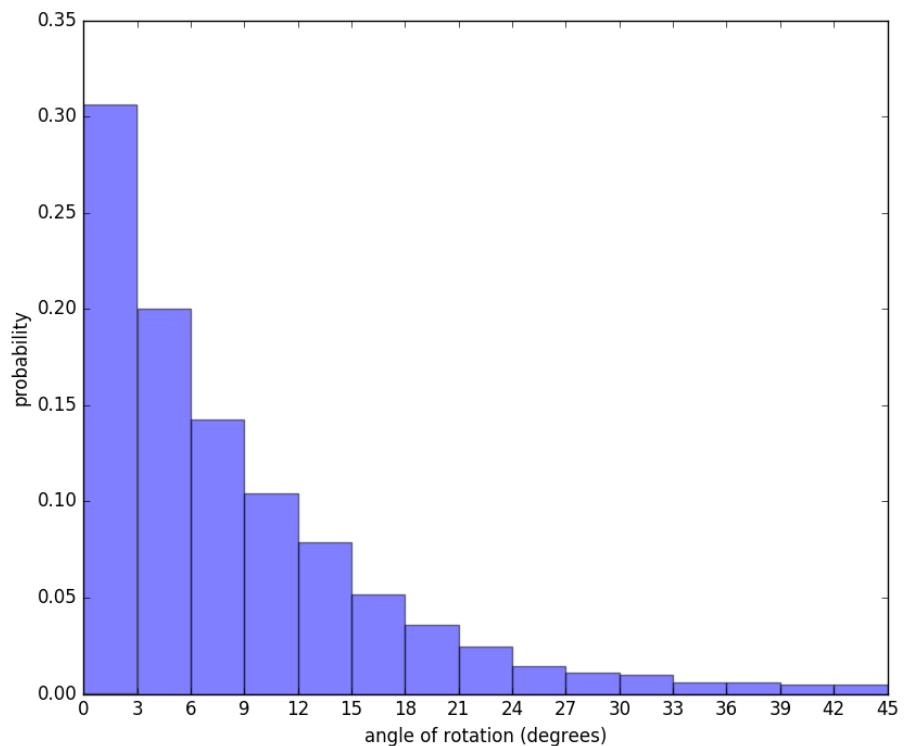


Figura 2.2: Angoli di rotazione delle bounding box in Incidental Text Scene.

2.1.3 COCO Text

Introdotto nel 2016 [12] come estensione del MSCOCO Dataset, viene in seguito utilizzato per l'omonima competizione in occasione di ICDAR2017. Punti di forza di questo dataset, oltre alla grande quantità di immagini, sono la varietà di contesti di acquisizione e la eterogeneità delle classi delle annotazioni. Per questo motivo lo si è scelto come base di training per i nostri esperimenti e come base per la validazione delle performance. Non ci è possibile confrontarci con metodi concorrenti in quanto i risultati della competizione non sono ancora stati resi pubblici.

Caratteristiche:

- 43.686 immagini di training (118.309 parole)
- 10.000 immagini di validation (27.550 parole)
- 10.000 immagini di test
- groundtruth composta da bounding box ortogonali e trascrizione per ogni parola, ognuna con i seguenti attributi:
 - machine printed / handwritten / altro
 - leggibile / illeggibile
 - inglese / non-inglese / sconosciuto

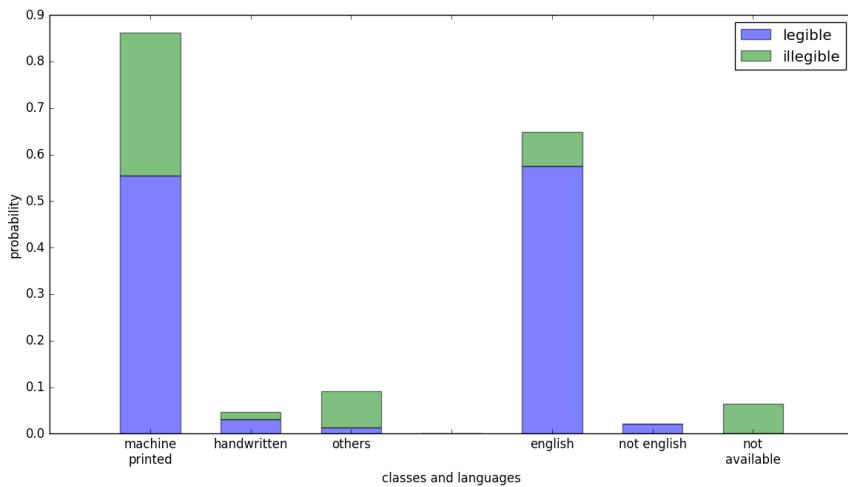


Figura 2.3: Classificazione delle annotazioni

2.1.4 SynthText

Introdotto nel 2016 [13], ma mai utilizzato come base di training in nessuna competizione, la peculiarità di questo dataset risiede nel fatto che il testo è inserito sinteticamente, con una certa context-awareness, in immagini reali.

Caratteristiche:

- 858.750 immagini
- 7.266.866 parole
- 28.971.487 lettere
- groundtruth composta da trascrizioni e coordinate dei vertici delle bounding box orientate di ogni parola ed ogni singola lettera



Figura 2.4: Immagine di training SynthText

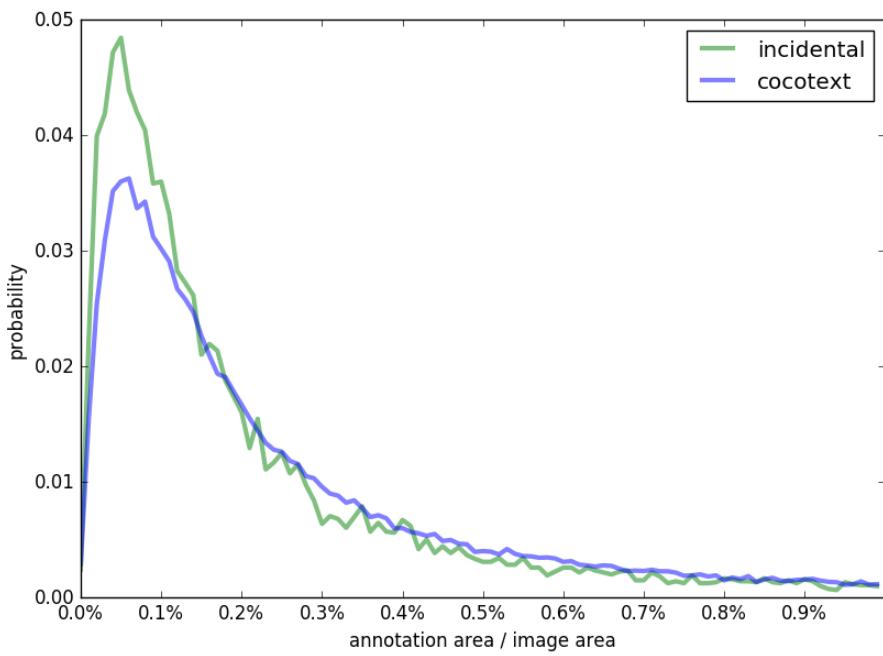


Figura 2.5: Area occupata dalle singole annotazioni nelle rispettive immagini

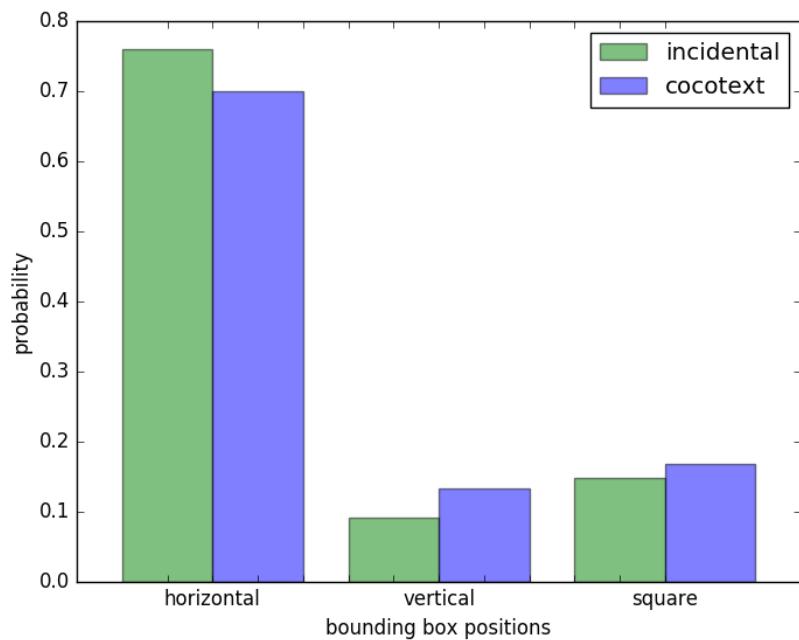


Figura 2.6: Proporzioni delle upright bounding box

2.2 Metodi di valutazione

Come già precedentemente annoverato, per la comparazione tra metodi diversi che concorrono per lo stesso fine, si necessita di valori di riferimento omogenei e significativi. In seguito verranno dunque illustrati più in dettaglio i vari metri di valutazione presenti in letteratura utilizzati per i nostri esperimenti e competizioni di riferimento.

2.2.1 Precision & Recall

Quelle che seguono in questa sezione sono statistiche generali per la classificazione binaria indipendenti dallo specifico task.

Precision

Indica la percentuale di predizioni corrette rispetto al numero di predizioni totali. Viene calcolata come segue:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} = \frac{tp}{n}$$

Recall

Indica la percentuale di predizioni corrette rispetto al numero di istanze significative totali. Viene calcolata come segue:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

F1-score

Questa misura prende in considerazione entrambi i criteri precedentemente descritti calcolandone la media armonica, evitando perciò i problemi che potrebbero nascere dalla semplice media aritmetica qualora la recall fosse molto bassa e la precision alta (poche predizioni ma corrette) e viceversa (molte predizioni fra cui molte erronee).

$$F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Average Precision

Anche questa stima utilizza sia la precision che la recall, ma in più necessita di un livello di confidenza per ogni predizione in modo da poter calcolare una curva di precision/recall dalla quale estrarre l'area sottostante, il cui valore corrisponde alla precisione media. Quantizzata risulta calcolabile come:

$$\sum_{k=1}^N P(k) \Delta r(k)$$

dove N è il numero totale di predizioni, $P(k)$ è la precisione al livello di confidenza k e $\Delta r(k)$ è la differenza tra i valori di recall tra gli step di confidenza $k - 1$ e k .

2.2.2 Intersection over Union

Se prima abbiamo visto metri di valutazione generici, quello qui descritto è invece specifico del campo dell'object detection. Infatti poiché nei task di rilevamento risulta altamente improbabile predire un match esatto di una bounding box, è nata la necessità di uno standard di valutazione coerente ed efficace con il quale si potesse distinguere un *vero positivo* da un *falso positivo* influenzando di conseguenza tutte le misure descritte fin'ora.

Come suggerito dal nome, questa metrica è calcolata prendendo il rapporto tra l'area dell'intersezione e l'area dell'unione di due bounding box, predizione e groundtruth. Essendo il risultato un semplice valore numerico, è necessario porre una soglia per l'avvenuta detection, solitamente impostata a 0.5 (oppure 0.75 per valutazioni più meticolose).

Una delle caratteristiche principali dell'IoU è la penalizzazione dei casi in cui oggetti multipli appartengono alla stessa classe vengano rilevati in una bounding box unica, ma specularmente si rischia di avere risultati ingannevoli nel caso in cui le annotazioni presentino errori o qualora il modello riesca addirittura a superare l'accurezza della groundtruth.

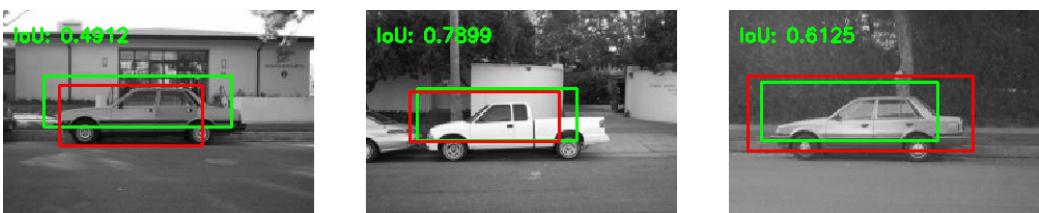


Figura 2.7: Esempi di valutazione con IoU.

Capitolo 3

Reti Neurali

In questo capitolo verrà illustrato lo strumento principale utilizzato per lo sviluppo di un metodo di text localization, partendo dapprima dagli albori delle reti neurali, passando per i vari sviluppi negli anni nei vari campi, fino ad arrivare alla struttura alla base del nostro metodo.

3.1 Reti Neurali Artificiali

Le *reti neurali artificiali* possono essere definite come un modello matematico ispirato alle reti neurali biologiche, costituito da neuroni artificiali interconnessi e da processi che dettano il cambiamento della propria struttura in base a informazioni che scorrono attraverso la rete durante la cosiddetta fase di apprendimento.

Il primo accenno al termine si deve a McCulloch e Pitts [14] che nel 1943 introdussero un combinatore lineare a soglia, esemplificato nella figura 3.1, in grado di ricreare funzioni booleane.

$$TLU(\vec{x}) = \phi\left(\sum_{i=1}^n w_i \cdot x_i\right) \quad n = 2 \quad w_1 = 1.0 \\ \phi(x) = x - \theta \geq 0 \quad \theta = 1.5 \quad w_2 = 1.0$$

Figura 3.1: Threshold Logic Unit con relativi parametri per l'implementazione dell'and logico.

Questo però non era che un semplice modello statico incapace di apprendere. L'intuizione che diede vita ad una gran varietà di studi riguardando metodi di apprendimento automatico si deve infatti a D.O. Hebb [15] il quale nel 1949 ipotizzò come l'apprendimento negli esseri viventi non fosse altro che un meccanismo di plasticità sinaptica, ovvero una modifica di intensità delle relazioni fra neuroni in base agli impulsi che li attraversano.

3.1.1 Perceptron

F. Rosenblatt ideò nel 1958 il *Perceptron* [16], un classificatore binario lineare, oggi considerato la più semplice rete neurale *feed-forward*.

Il Perceptron può essere modellato come segue:

$$y = f(x) = \begin{cases} 1 & \text{se } w\vec{x} > 0 \\ 0 & \text{altrimenti} \end{cases}$$

dove si incorpora un valore scalare di *bias* con

$$w = [w_1 \quad \cdots \quad w_d \quad b] \quad \vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \\ 1 \end{bmatrix}$$

Si parte dunque da un dataset $D = \{(x^{(1)}, g^{(1)}), \dots, (x^{(n)}, g^{(n)})\}$ formato da n campioni $x^{(i)} \in \mathbb{R}^d$, vettori delle feature d -dimensionali, e $g^{(i)}$ valore di output desiderato per $f(x^{(i)})$. L'algoritmo è descritto di seguito:

1. Inizializzazione del vettore w con valori nulli o randomici tendenti a zero.
2. Per ogni campione i nel dataset D :
 - (a) Calcolare y al tempo corrente t :

$$y^{(i)}(t) = f[w(t) \cdot \vec{x}^{(i)}]$$
 - (b) Aggiornare i valori di w al nuovo tempo $t + 1$:

$$\forall j \quad w_j(t + 1) = w_j(t) + (g^{(i)} - y^{(i)}(t))x_j^{(i)}$$
 - (c) Ripetere dal punto 2 per minimizzare $\frac{1}{n} \sum_{i=1}^n |g^{(i)} - y^{(i)}(t)|$

3.1.2 Backpropagation

Dal 1969 la ricerca si bloccò con la dimostrazione da parte di Minsky e Papert [17] dell'impossibilità di risolvere problemi con soluzioni non separabili linearmente. Negli anni a seguire le reti neurali vennero presto dimenticate a favore di altri modelli, come le Support Vector Machine, al tempo più efficaci nell'ambito del machine learning.

Nonostante questo nel 1975 venne introdotto da parte di Paul Werbos l'algoritmo per la cosiddetta *backpropagation* il quale rimane tutt'oggi un ingranaggio principale per il funzionamento della gran parte delle reti neurali.

Sia N una rete neurale con e connessioni, m input, n output, p osservazioni ed E una funzione di costo da ottimizzare. Avremo dunque:

$$\left. \begin{array}{ll} x^{(1)}, \dots, x^{(p)} & \text{vettori di input in } \mathbb{R}^m \\ y^{(1)}, \dots, y^{(p)} & \text{vettori di output in } \mathbb{R}^n \\ w^{(0)}, \dots, w^{(p)} & \text{vettori dei pesi in } \mathbb{R}^e \end{array} \right\} \begin{array}{l} y = f_N(x, w) \\ E : \mathbb{R}^{n \times n} \rightarrow \mathbb{R} \end{array}$$

L'output dell'algoritmo è dato da un nuovo vettore dei pesi ottenuto con il metodo del gradiente, calcolato di conseguenza dalla derivata della funzione di costo rispetto a w :

$$\frac{dE}{dw}$$

Per una serie di input perciò, partendo dal vettore $w^{(0)}$ inizializzato con valori prossimi a zero, si calcola $E(y^{(i)}, f(x^{(i)}, w^{(i-1)}))$ iterativamente ponendo $w^{(i)}$ uguale all'output dell'algoritmo alla fine di ogni iterazione.

3.1.3 Multilayer Perceptron

Partendo dai concetti precedentemente presentati passeremo ora a mostrare come costruire una semplice rete neurale artificiale con la relativa fase di addestramento: il *Multilayer Perceptron*.

Esso è così chiamato in quanto non è altro che un'estensione del Perceptron che può essere visto come una rete neurale con d neuroni in entrata (rappresentanti il vettore $x \in \mathbb{R}^d$) ed un singolo neurone in uscita, con le connessioni rappresentate dalla funzione $f(x)$.

Nel Multilayer Perceptron questa struttura viene ampliata aggiungendo almeno uno *strato nascosto* tra il layer di input e quello di output formato da

un numero arbitrario di neuroni. Ciò che si viene a creare è dunque un grafo diretto aciclico, come mostrato in figura 3.2, in cui i neuroni appartenenti allo stesso layer non sono connessi fra di loro.

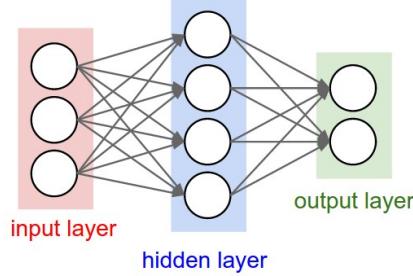


Figura 3.2: Rappresentazione grafica di un Multilayer Perceptron

Le connessioni fra due strati x e y di n e m neuroni rispettivamente, possono essere formalizzate come segue:

$$\left. \begin{array}{l} x \in \mathbb{R}^{n \times 1} \\ W \in \mathbb{R}^{m \times n} \end{array} \quad \begin{array}{l} y \in \mathbb{R}^{m \times 1} \\ b \in \mathbb{R}^{m \times 1} \end{array} \right\} y = \phi(Wx + b)$$

Altra aggiunta che si può notare nella formalizzazione è la presenza di ϕ ovvero una funzione di attivazione che aggiunge non linearità al modello multistrato e permette alla rete di approssimare qualsiasi funzione, come dimostrato nel 1989 da Cybenko [18]. Le funzioni di attivazione, illustrate nella figura 3.3, possono essere di vario tipo, di seguito sono elencate le più importanti:

Funzione Logistica	$(1 + e^{-x})^{-1}$
Tangente Iperbolica	$\tanh(x)$
Rectified Linear Unit (ReLU)	$\max(0, x)$
Leaky ReLU	$\max(0.01x, x)$
LReLU Parametrica	$\begin{cases} \alpha x & \text{se } x \geq 0 \\ x & \text{se } x < 0 \end{cases}$
Exponential Linear Unit (ELU)	$\begin{cases} \alpha(e^x - 1) & \text{se } x \geq 0 \\ x & \text{se } x < 0 \end{cases}$

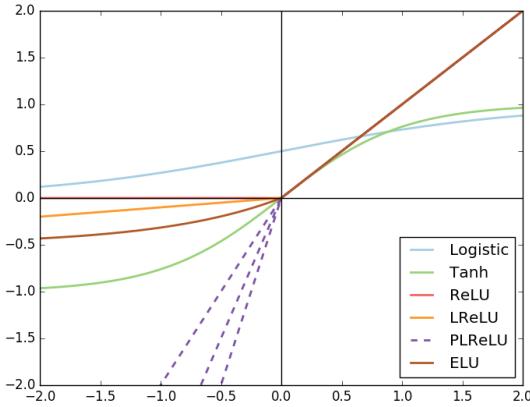


Figura 3.3: Comparazione fra le diverse funzioni di attivazione.

Dato il modello appena descritto dunque non rimane che utilizzarlo come classificatore (con più di due classi e non-lineare a differenza del Perceptron di base) oppure per un task di regressione. Il prossimo passo è perciò quello di definire la funzione di costo (detta in gergo di *Loss*) da utilizzare con l'algoritmo di backpropagation.

3.1.4 Funzioni di Loss

Classificazione

Assumiamo che ogni neurone f_j del layer di output rappresenti uno score di appartenenza ad un certa classe e y sia la classe a cui appartiene x :

Hinge Loss

Δ rappresenta la distanza minima richiesta di ogni score da quello della classe corretta.

$$L = \sum_{j \neq y} \max(0, f_j - f_y + \Delta)$$

Softmax Cross-Entropy

Con questo metodo lo score di ogni classe viene normalizzato. Il valore di loss finale è tanto minore quanto vicino a 1 è la probabilità della classe corretta.

$$L = -\log \left(\frac{e^{f_y}}{\sum_j e^{f_j}} \right)$$

Regressione

Assumiamo che ogni neurone f_i del layer di output rappresenti una quantità da predire e y sia un vettore dei valori attesi.

L1 Loss

Distanza di Manhattan.

$$L = \|f - y\|_1 = \sum_i |f_i - y_i|$$

L2 Loss

Distanza euclidea al quadrato per facilitare il calcolo della derivata durante la backpropagation.

$$L = \|f - y\|_2^2 = \sum_i (f_i - y_i)^2$$

3.1.5 Apprendimento

Ultimo passo per completare il meccanismo di apprendimento di una rete neurale è quello di definire come l'algoritmo di backpropagation dovrà aggiornare i valori dei vari neuroni.

Il metodo base della discesa stocastica del gradiente (SGD) opera come segue su un certo vettore x di parametri al tempo $t + 1$ utilizzando un *learning rate* η predefinito:

$$x_{t+1} = x_t - \eta \cdot \frac{dL}{dx_t}$$

In seguito verranno elencate diverse ottimizzazioni da poter usare insieme al metodo di base per accellerare il processo di apprendimento.

Ci riferiremo ai gradienti calcolati più semplicemente con:

$$dx_t = \frac{dL}{dx_t}$$

Momentum Update

[19] Viene aggiunto un ulteriore iperparametro α e una variabile Δx in maniera da favorire la discesa in punti con alto gradiente.

$$\begin{aligned}\Delta x_{t+1} &= \eta \cdot dx_t + \alpha \Delta x_t \\ x_{t+1} &= x_t - \Delta x_{t+1}\end{aligned}$$

Adagrad

[20] Learning rate adattivo per ogni singolo parametro grazie ad una variabile che tiene traccia del quadrato del gradiente.

$$G_x^{(t+1)} = G_x^t + dx_t^2$$

$$x_{t+1} = x_t - \frac{\eta \cdot dx_t}{\sqrt{G_x^{(t+1)}} + \epsilon}$$

RMSProp

[21] Simile ad Adagrad con l'aggiunta di un iperparametro β (decay rate) che previene la descrescenza monotonica del learning rate mantenendo una media mobile del quadrato del gradiente.

$$G_x^{(t+1)} = \beta \cdot G_x^t + (1 - \beta) \cdot dx_t^2$$

$$x_{t+1} = x_t - \frac{\eta \cdot dx_t}{\sqrt{G_x^{(t+1)}} + \epsilon}$$

Adam

[22] Utilizza le medie mobili sia del gradiente che del suo quadrato introducendo dunque due iperparametri β_1 e β_2 . Queste vengono in seguito corrette per evitare un annichilimento durante le prime iterazioni.

$$m_x^{(t+1)} = \beta_1 \cdot m_t + (1 - \beta_1) \cdot dx_t \quad \hat{m}_x = \frac{m_x^{(t+1)}}{1 - \beta_1^t}$$

$$v_x^{(t+1)} = \beta_2 \cdot v_t + (1 - \beta_2) \cdot dx_t^2 \quad \hat{v}_x = \frac{m_x^{(t+1)}}{1 - \beta_2^t}$$

$$x_{t+1} = x_t - \frac{\eta \cdot \hat{m}_x}{\sqrt{\hat{v}_x} + \epsilon}$$

3.1.6 Ottimizzazioni

In questa sezione verranno discussi i principali metodi che consentono il miglioramento delle prestazioni di una qualsiasi rete neurale, sia dal punto di vista della velocità che della qualità dell'apprendimento.

Data preprocessing

Le principali operazioni sono quelle di sottrazione del *valore medio* dall'intero campione di dati di training e la sua *normalizzazione* attraverso la divisione per la deviazione standard. Queste operazioni permettono di avere dei dati geometricamente centrati all'origine e con valori compresi tra -1 e 1.

Dropout

Introdotto nel 2014 [23], consiste nell'introdurre una probabilità p che ogni neurone ha di rimanere attivo durante la fase di addestramento. L'obiettivo è quello di evitare l'overfitting dei dati attraverso epoch diverse.

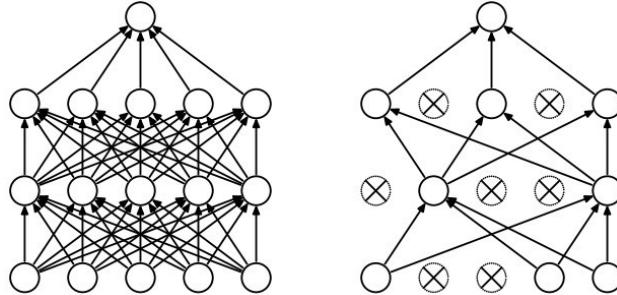


Figura 3.4: Rete neurale prima e dopo l'applicazione del dropout.

Inizializzazione dei pesi

Dato che un'inizializzazione costante di tutti i pesi porterebbe a gradienti costanti, una soluzione consiste nell'utilizzare valori randomici prossimi a 0 distribuiti secondo una normale divisi per la radice quadrata del numero di input del layer. Ciò è dovuto dal fatto che un numero maggiore porterebbe ad una varianza sempre maggiore. Dato però l'utilizzo di funzioni di attivazione, viene violata l'assunzione di una media nulla ed è dunque stata identificato il valore $\sqrt{2/n}$ come reale valore della varianza [24].

Batch Normalization

Introdotta nel 2015 [25] con l'obiettivo di contrastare inizializzazioni sfavorevoli dei pesi nella rete, consiste nell'inserimento di un layer di normalizzazione tra ogni componente connessa e la relativa funzione di attivazione.

3.2 Reti Neurali Convoluzionali

Il modello di rete neurale visto fino ad ora rappresenta solamente le fondamenta degli strumenti odierni. In seguito verranno dunque descritte le cosiddette *CNN* in quanto modello principale che ha preso piede nel campo della Computer Vision.

3.2.1 Struttura delle CNN

Prima di illustrare le varie architetture che si son succedute negli anni è doveroso spiegare come si possa adattare un modello basato su *fully-connected layers* (come per il Multilayer Perceptron visto nella sezione 3.1.3) in modo da avere come input delle immagini (anche di dimensione diversa) evitando l'esplosione del numero di pesi da addestrare.

Convolution Layer

Ovvero il layer principale che dà il nome alle reti neurali convoluzionali. L'idea principale è quella di avere n filtri, solitamente di 3 o 5 pixel per lato, che vengono applicati all'input. Gli iperparametri principali sono:

- **Dimensione** del filtro.
- **Profondità**, il numero di filtri.
- **Stride**, di quanti pixel spostare il filtro.
- **Padding**, se aggiungere un bordo all'input per mantenere costante la dimensioni in output e nel caso quali valori usare.

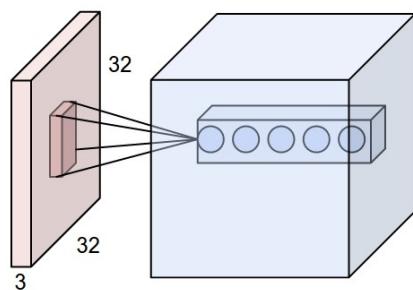


Figura 3.5: Visualizzazione di un'operazione di convoluzione.

ReLU Layer

Applicazione delle funzioni di attivazione descritte nella sezione 3.1.3.

Pooling Layer

Consente di ridurre la dimensione spaziale dell'input in base ad iperparametri s stride e n estensione del campo ricettivo.

I criteri di scelta del valore di output sono principalmente due: *max pooling* (scelta del massimo) oppure *average pooling* (scelta della media).

Deconvolution Layer

Non è altro che una convoluzione con stride frazionario che aumenta le dimensioni spaziali (de-pooling) dell'input ma che ne riduce la profondità invertendo dunque il *forward pass* con il *backward pass*.

3.2.2 Breve storia

LeNet

Nonostante già nel 1988 [26] vennero mossi i primi passi verso l'ideazione di un precursore delle reti convoluzionali, solo l'architettura ideata nel 1998 da LeCun et al. [27] viene oggi considerata come la prima vera *CNN*.

Questo modello, introdotto con lo scopo di riconoscere caratteri scritti a mano, come mostrato in figura 3.6, era composto da 4 sequenze di convoluzioni e ridimensionamenti, seguiti da 3 fully-connected layer ed una funzione a base radiale gaussiana finale per stimare gli errori per le singole classi.

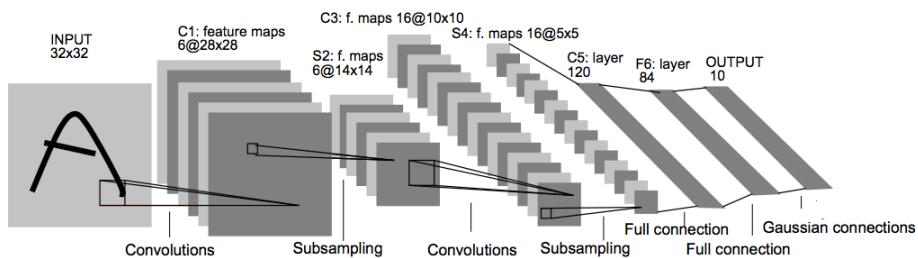


Figura 3.6: Schema di LeNet-5.

GPGPU

Questi sviluppi furono ancora osteggiati dall'elevata capacità computazionale richiesta per l'addestramento delle reti neurali. Nei primi anni del nuovo millennio infatti gli studi si concentrarono principalmente sullo studio di tecniche di addestramento attraverso l'utilizzo di *Graphic Compute Units* in seguito alla pubblicazione di una ricerca nel 2005 che ne descriveva il loro valore nel campo del machine learning. [28]

3.2.3 Casi di studio

Di seguito descriveremo velocemente diverse architetture sviluppatisi molto recentemente in seguito all'implementazione di LeNet su GPU nel 2011. [29]

AlexNet

Questa rete neurale [30] è presentata in occasione della *ImageNet ILSVRC Challenge* [31] nel 2012, competizione annuale che prevedeva task di classificazione e localizzazione di ben 1000 classi differenti. Prendendo spunto direttamente da LeNet, questa architettura è riuscita a raggiungere risultati inimmaginabili sovraperformando il secondo classificato di ben 10 punti sulla percentuale d'errore.

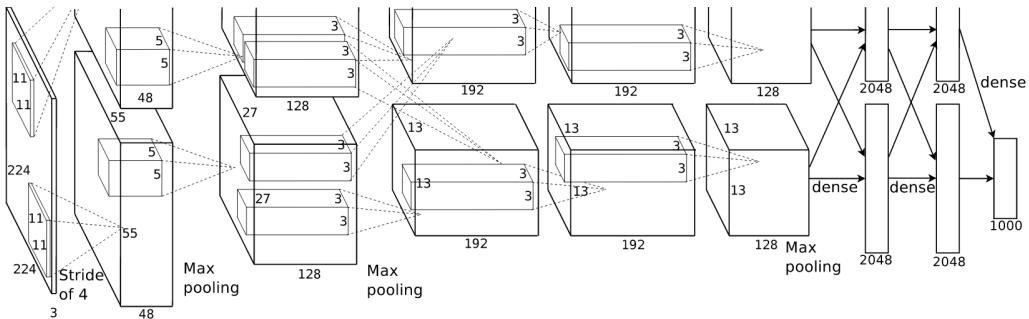


Figura 3.7: Schema di AlexNet.

GoogLeNet

Vincitrice della *ImageNet ILSVRC Challenge* nel 2014, ideata da Szegedy et al. da Google [32], ha introdotto il concetto di *Inception Module*, illustrato in figura 3.8, in maniera da ridurre drasticamente il numero di parametri (4 milioni rispetto ai 60 di AlexNet).

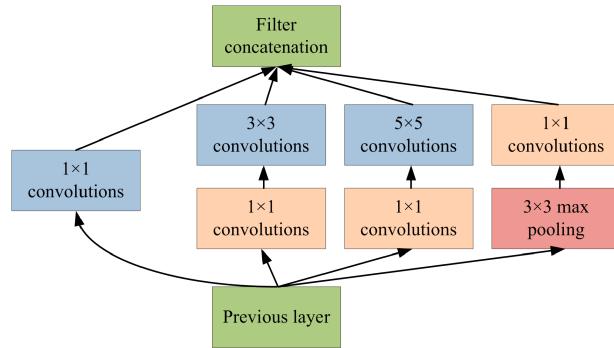


Figura 3.8: Inception Module di GoogLeNet.

VGG

Nonostante si è classificata solo in seconda posizione scontrandosi con GoogLeNet, questa architettura ideata da Karen Simonyan and Andrew Zisserman [33] ha mostrato che la *profondità* di una rete era una componente principale per performance maggiori. Altro punto di forza è la sua semplicità strutturale nonostante il numero di parametri elevato, ben 140 milioni, dovuti agli ultimi fully-connected layer.

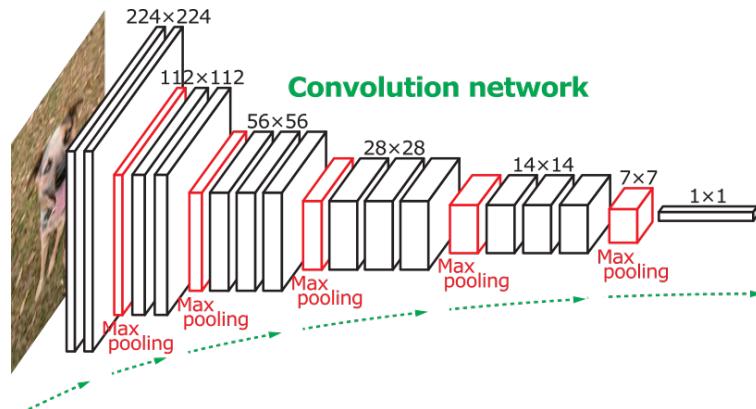


Figura 3.9: Schema di VGG16.

ResNet

[34] Vincitrice di *ILSVRC Challenge 2015*, introduce le cosiddette *Shortcut Connections*, illustrate in figura 3.10, in modo da risolvere il problema della degradazione del segnale di errore in reti molto profonde.

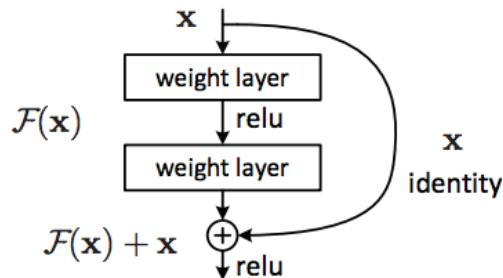


Figura 3.10: Shortcut Connection.

3.3 Fully Convolutional Network

Tutte le architetture citate in precedenza per quanto potenti ed innovative, nella loro configurazione di base sono solo atte a classificare un’immagine, ovvero a fornire le probabilità di appartenenza a ciascuna classe su cui sono state addestrate.

Questa limitazione perciò non ci permette di eseguire classificazioni multiple o localizzazioni, per esempio separare due oggetti appartenenti alla stessa classe oppure rilevare due oggetti di due classi differenti all’interno di una stessa immagine.

Per la localizzazione singola di una classe una soluzione banale è quella di passare da un task di classificazione ad uno di regressione delle coordinate della bounding box cambiando dunque la funzione di loss finale.

Negli anni si sono avuti ulteriori raffinamenti atti a compensare queste lacune, dapprima con FastRCNN [35] ed il successivo FasterRCNN [36] e più recentemente con YOLO [37], tutti nati con lo scopo di predire bounding box multiple con rispettive classi di appartenenza. Quello però di cui tratteremo in questa sezione è un approccio completamente differente.

Ideata nel 2014 [38], una *Fully Convolutional Network* si pone l’obiettivo di generare un output della stessa dimensione dell’immagine di input in

cui ogni pixel rappresenta la propria classe di appartenenza, praticamente semgentando l'immagine originale.

Grande pregio di questa architettura è la propria semplicità strutturale, infatti, come mostrato in figura 3.11, non fa altro che riprendere la *VGG16* ed ampliarla aggiungendo uno o più layer di deconvoluzione addizionati ai precedenti output dei layer di pooling per generare l'immagine finale.

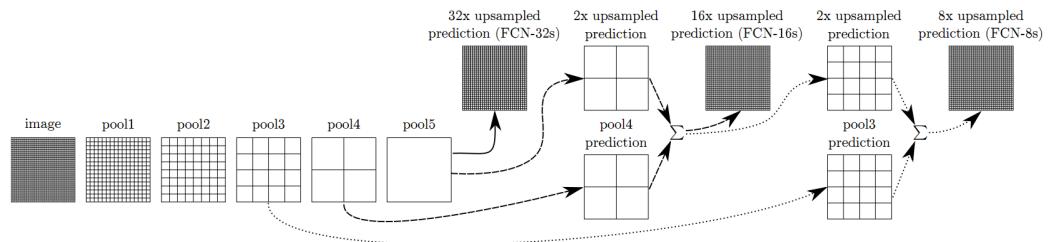


Figura 3.11: Schema delle Fully Convolutional Network.

Capitolo 4

Esperimenti

Per il perseguitamento dello scopo di questa tesi si è deciso di utilizzare un approccio con reti neurali convoluzionali basato sull'architettura delle Fully Convolutional Networks esposte nel capitolo precedente. Verranno dunque qui mostrate le motivazioni di questa scelta, l'evoluzione del metodo sviluppato e varie considerazioni sui risultati ottenuti.

4.1 Descrizione del metodo

Recentemente, come già accennato, l'architettura delle FCN si è dimostrata un mezzo potente per il rilevamento di oggetti nelle immagini, rendendo superflua la fase di *region proposal* e ponendo in ombra metodi per la localizzazione multipla basati su RRCN.

È naturale quindi porsi la domanda di dove può arrivare la potenza di queste reti per task diversi da quelli per cui sono state concepite, ricordando che alla base vi è un generico classificatore.

Il nostro obiettivo è quello di trovare aree di testo in immagini digitali, quindi qualcosa che va al di là della semplice concezione di *oggetto*. Un testo può infatti essere presente su una varietà immensa di oggetti eterogenei, basti pensare ai cartelli stradali, ai giornali, alle magliette oppure agli schermi di televisori e smartphone.

Pare dunque impossibile definire un oggetto singolo classificabile come testo, se non pensando di isolare singolo carattere, obiettivo che ci viene reso impossibile dall'indisponibilità di un dataset adeguato.

Nonostante tutto, invece questa concezione astratta può venire superata dalle reti neurali attraverso un corretto addestramento date le giuste informazioni. A prova di ciò possiamo citare [39], dove per localizzare le singole istanze di vari oggetti vengono utilizzate due FCN aggiuntive, una per la predizione della profondità e l'altra per la direzione dal centroide di ogni istanza. Si veda l'immagine 4.1 per un'esempio della groundtruth.



Figura 4.1: Esempi delle groundtruth.

I più che discreti risultati di questo lavoro ci dimostrano dunque la grande versatilità di una FCN per un grande campo di mansioni che esulano da quello della loro concezione originale.

Assodato il fatto che verrà utilizzata una FCN per la segmentazione di un'immagine in area di testo e non, di seguito verrà descritto il processo di sviluppo della nostra pipeline per la localizzazione per poi confrontarci con metodi concorrenti.

4.1.1 Strumenti

Python [40] si è imposto negli anni come linguaggio molto semplice da usare per la prototipizzazione di algoritmi nonostante la sua lentezza dovuta alla sua natura di linguaggio interpretato. Questa sua debolezza è però stata in parte superata grazie a moduli sviluppati come *wrapper* intorno a librerie scritte in linguaggi compilati come C/C++, rendendo dunque più accessibile e semplificato l'utilizzo di routine altamente efficienti.

Tra le più famose, ed utilizzate in questa tesi, citiamo **NumPy** [41] per la manipolazione di array multidimensionali, **SciPy** [42] per gli algoritmi matematici più vari, **OpenCV** [43] per l'elaborazione di immagini ed infine **Shapely** [44] per la manipolazione di oggetti geometrici.

Per l'implementazione della *Fully Convolutional Network* però ci si è affidati a **TensorFlow** [45], una libreria per l'implementazione, visualizzazione e profilazione di reti neurali sviluppata da Google dal 2015.

4.1.2 Groundtruth

L'output desiderato della nostra rete consiste in un'immagine delle stesse dimensioni di quella di input segmentata in due classi: testo e “non testo”, quindi come prima idea potrebbe sovvenire quella di specificare, lettera per lettera, le aree di testo rispetto al background. Ciò potrebbe sembrare l'approccio migliore atto a generare risultati accuratissimi non solo per la localizzazione, ma anche per la segmentazione delle singole lettere per il successivo task di riconoscimento del testo, ma ad oggi non esistono dataset simili a nostra disposizione.

Il passo successivo è quindi quello di generalizzare l'idea precedente portando l'area di interesse non quindi alla forma della singola lettera, ma la sua posizione. Il dataset SynthText introdotto nella sezione 2.1.4 viene in nostro soccorso, essendo infatti un dataset sintetico ci mette a disposizione questi dati specifici fornendo le bounding box per ogni carattere.

Per quanto riguarda gli altri dataset introdotti in precedenza, le loro annotazioni ci forniscono solo i dati approssimativi di una bounding box per parola, quindi possiamo ancora astrarre e considerare quelle zone come area di interesse, seppur aggiungendo teoricamente molto più rumore.

4.1.3 Iperparametri

La scelta di iperparametri per l'ottimizzazione di una rete neurale è una delle sfide maggiori che si affrontano durante la fase di addestramento di un modello. Di seguito sono elencati quelli principali individuati durante le fasi preliminari di preparazione della rete:

Batch size: Un numero maggiore velocizza l'apprendimento. 16 è il massimo che si è potuto raggiungere rimanendo nei limiti di memoria a disposizione.

Learning rate: Influenzato dal batch size, nonostante l'utilizzo di algoritmi sofisticati per la discesa del gradiente, valori troppo grandi (10^{-4}) non portano a nessuna predizione, mentre valori troppo piccoli (10^{-6}) rallentano l'apprendimento. 10^{-5} è il valore utilizzato per tutti gli esperimenti a seguire con Adam.

Loss function: Pixelwise Softmax Cross-Entropy con azzeramento del valore di loss su zone con testo contrassegnato come illeggibile dai dataset in uso.

4.1.4 Dettagli preliminari

Prima di procedere con l'analisi dei vari esperimenti svolti è necessario chiarire alcuni punti riguardanti implementazione della rete, gestione dell'apprendimento, estrazione delle predizioni finali e utilizzo dei vari dataset.

FCN

Qui elencate le caratteristiche principali dell'architettura:

- *FCN-8* (3 deconvoluzioni con addizione di due layer di pooling come illustrato in figura 3.11)
- *VGG19* come classificatore con pesi già addestrati per ImageNet [46]
- *ReLU* standard
- *Max Pooling*
- *Padding* nell'ultima convoluzione della VGG per facilitare l'utilizzo della rete neurale su immagini di dimensione variabile

Generazione dati di training

Poiché le immagini dei vari dataset presentano risoluzioni pressoché sempre differenti, è necessario definire un metodo per ottenere zone di grandezza fissa da ciascuna di esse in modo da adoperare un batch size maggiore di 1.

L'idea di estrarre una zona randomica, seppur molto semplice, potrebbe rallentare l'addestramento in quanto, come già analizzato in figura 2.5, le zone di testo occupano un'area decisamente esigua rispetto all'immagine in cui compaiono, abbassando dunque le probabilità che possano essere incluse nel nostro "ritaglio" randomico.

La soluzione adottata consiste dunque nel partire da un'annotazione contrassegnata come leggibile e da lì espandere randomica la selezione verso i bordi dell'immagine, squadrarla (anche uscendo dai limiti) ed infine ridimensionare l'area selezionata alla risoluzione prefissata.

Questo approccio ha un grande pregio, infatti per ogni immagine di partenza potrebbero essere create decine di esempi differenti, estendendo di conseguenza il dataset di base.

L'algoritmo sviluppato è presentato nello pseudocodice 1.

Algorithm 1 Generazione dati di training

```
function GETIMAGE(img, annGt, weightGt, id, cropSize)
    annotation  $\leftarrow$  RANDOM(ANNOTATIONS(id))
     $\triangleright$  Espandi la bounding box verso i bordi.
     $x_1, x_2, y_1, y_2 \leftarrow annotation.bbox$ 
     $x_1 \leftarrow x_1 - \text{RANDOM}(0 \dots x_1)$ 
     $y_1 \leftarrow y_1 - \text{RANDOM}(0 \dots y_1)$ 
     $x_2 \leftarrow x_2 + \text{RANDOM}(0 \dots img.width - x_2)$ 
     $y_2 \leftarrow y_2 + \text{RANDOM}(0 \dots img.height - y_2)$ 
     $ratio \leftarrow (x_2 - x_1) / (y_2 - y_1)$ 
     $\triangleright$  Modifica le coordinate in modo da ottenere un quadrato.
    if ratio > 1 then
         $diff \leftarrow (x_2 - x_1) - (y_2 - y_1)$ 
         $split \leftarrow \text{RANDOM}(0 \dots diff)$ 
         $y_1 \leftarrow y_1 - split$ 
         $y_2 \leftarrow y_2 + (diff - split)$ 
    else if ratio < 1 then
         $diff \leftarrow (y_2 - y_1) - (x_2 - x_1)$ 
         $split \leftarrow \text{RANDOM}(0 \dots diff)$ 
         $x_1 \leftarrow x_1 - split$ 
         $x_2 \leftarrow x_1 + (diff - split)$ 
     $\triangleright$  Ottieni i valori di padding nel caso si esca dai limiti.
     $pad_y \leftarrow (\max(0, -y_1), \max(0, y_2 - img.height))$ 
     $pad_x \leftarrow (\max(0, -x_1), \max(0, x_2 - img.width))$ 
     $x_1 \leftarrow \max(0, x_1)$ 
     $y_1 \leftarrow \max(0, y_1)$ 
     $\triangleright$  Per ogni immagine (originale, groundtruth e pesi)
     $\triangleright$  ritaglia, aggiungi padding e applica resize.
    for all element  $\in [img, annGt, weightGt]$  do
        element  $\leftarrow element[y_1 \dots y_2][x_1 \dots x_2]$ 
        element  $\leftarrow \text{PAD}(\text{element}, pad_x, pad_y)$ 
        element  $\leftarrow \text{RESIZE}(\text{element}, cropSize)$ 
    return img, annGt, weightGt
```

Predizioni

Partendo dal presupposto che il risultato finale su cui operare sia un’immagine binaria in cui le zone “attive” rappresentano una predizione di un’area di testo, ciò che si andrà a fare per estrarre le bounding box in breve è quanto segue:

- operazione morfologica di closing con una maschera 3×3
- labeling delle componenti connesse con maschera 3×3 completa
- eliminazione delle c.c. con un’area minore di una certa soglia
- estrazione della bounding box (ruotata o meno in base al test set) da ciascuna componente connessa

Lo pseudocodice 2 illustra l’algoritmo.

Algorithm 2 Estrazione delle bounding box da immagine binaria

```
function GETPREDICTIONS(img, threshold)
    img  $\leftarrow$  CLOSING(img)
     $\triangleright$  Trova le componenti connesse
    labeled, nlables  $\leftarrow$  LABEL(img)
    bboxes  $\leftarrow$   $\emptyset$ 
    if nlables  $> 0$  then
         $\triangleright$  Date le componenti connesse, ottieni le loro aree
        areas  $\leftarrow \{\|\text{labeled}[\text{labeled} = i]\| \mid i \text{ from } 1 \text{ to } \text{nlables}\}$ 
        for i from 1 to nlables do
            if areas[i]  $\geq$  threshold then
                lone  $\leftarrow$  copy labeled
                 $\triangleright$  Isola una componente connessa
                lone[lone  $\neq$  i]  $\leftarrow 0$ 
                cnt  $\leftarrow$  FINDCONTOUR(lone)
                 $\triangleright$  per estrarre la bounding box
                bboxes  $\leftarrow$  bboxes  $\cup$  {BOUNDINGBOX(cnt)}
    return bboxes
```

Dataset

Non è stato possibile utilizzare COCO-Text in fase di testing in quanto i risultati pubblici non sono ancora disponibili in data di stesura di questa tesi, mentre SynthText non dispone di sottoinsieme di testing prefissato.

Focused Scene Text è stato ignorato per la fase di training poiché presenta pochi dati che si possono facilmente ritrovare in Incidental Scene Text.

	Training	Validation ¹	Testing ²
COCO-Text	•	•	
SynthText	•		
Incidental Scene Text	•		•
Focused Scene Text			•

¹ Comparazione fra modelli addestrati diversamente

² Confronto finale con metodi di terzi

4.2 Addestramento sui dataset

4.2.1 COCO-Text

Questo dataset, nonostante le sue imprecisioni, comincia subito a produrre risultati visivamente coerenti e significativi, come mostrato nella seguente immagine 4.2.



Figura 4.2: Immagine originale e immagine combinata con mappa delle probabilità.

Osservando l'andamento della loss per tutte le iterazioni si può notare un trend che farebbe pensare ad un overfitting sui dati, infatti da 60mila iterazio-

ni in poi (circa 64 epoche¹) la loss su un sottoinsieme fisso del validation set rimane pressoché invariata mentre quella del training è in lenta discesa.

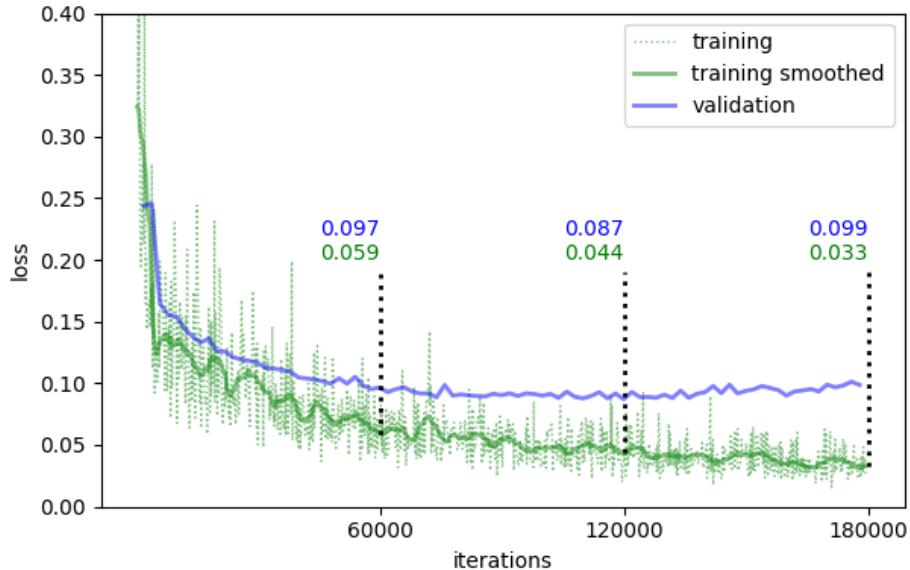


Figura 4.3: Curve di loss su training a validation set.

Applicando invece l'algoritmo di estrazione delle bounding box con soglia 32 pixel ed infine valutando con *Intersection over Union* a 0.5, i miglioramenti vi sono, come si può notare dai risultati riportati nella tabella 4.1.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illeggibili	Totale		
Intero validation set					
60k	16.21	4.14	12.28	13.85	13.01
180k	20.87	9.59	17.29	17.72	17.50
Almeno un leggibile per immagine					
60k	—	3.41	12.79	24.11	16.71
180k	—	9.00	17.78	28.15	21.79

Valori espressi in percentuale.

Tabella 4.1

¹nonostante parlare di epoche abbia poco senso in quanto, come già spiegato nella sezione 4.1.4, il training set è mutevole

4.2.2 Incidental Scene Text

I risultati già dopo le prime 60mila iterazioni (960 epoche) non sono promettenti, infatti si presenta un calo drastico nella precisione rispetto a COCO-Text. Si confrontino i risultati riportati nella tabella 4.2 con quelli della precedente tabella 4.1.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illegibili	Totale		
Intero validation set					
60k	19.07	5.78	14.54	7.87	10.21
Almeno un leggibile per immagine					
60k	—	5.69	15.48	13.95	14.67

Valori espressi in percentuale.

Tabella 4.2

Il problema potrebbe essere imputabile alla groundtruth lievemente differente rispetto a quella di COCO-Text la quale presenta poche bounding box ruotate. Più avanti ritorneremo ad analizzare questo fenomeno.

4.2.3 SynthText

L’addestramento su questo dataset viene presto abbandonato ancora in fase sperimentale in quanto i risultati visivi (si veda la figura 4.4) mostrano chiaramente un apprendimento poco significativo.



Figura 4.4

Il problema è molto probabilmente dovuto alla natura sintetica del dataset per cui molti elementi testuali, seppur posizionati tenendo conto della forma e perspettiva, si ritrovano in contesti completamente irreali e risultano dunque fuorvianti ai fini dell’addestramento della rete neurale.

4.3 Modifiche strutturali

I risultati ottenuti, seppur mostrino come questo approccio sia corretto nella forma, non raggiungono numeri soddisfacenti, l’obiettivo è dunque quello di cercare di migliorare senza però complicare il processo di estrazione delle bounding box illustrato nello pseudocodice 2. Ciò è dato dal fatto che il fine postoci è quello di adoperare principalmente una FCN, senza dunque ricorrere a metodi raffinati da porre in coda alla catena di elaborazione dei dati come effettuato da diversi altri metodi citati in precedenza nella sezione 1.2.

Osservando le predizioni ottenute con l’addestramento su COCO-Text, si può notare come le zone di testo predette non siano molto raffinate e finiscano dunque per raggruppare insieme molte parole vicine fra di loro. Per ovviare a questo problema le soluzioni adottate che andremo a descrivere sono principalmente due:

- Nuova classe “bounding box” che rappresenti il contorno di una parola.
- Nuova classe “separazione” che rappresenti una linea immaginaria che separi due parole vicine fra di loro.

In quanto difficilmente una *Fully Convolutional Network* è in grado di generare una segmentazione raffinata in presenza di più classi, soprattutto quando queste sono simili o complementari, la generazione delle predizioni di aree testuali verrà effettuata come descritto nell’algoritmo 3, piuttosto che nel modo tradizionale con il quale si sceglierrebbe la classe con score maggiore.

Algorithm 3

```

 $output \leftarrow \emptyset \in \mathbb{R}^{n \times m}$                                  $\triangleright probText, probThird \in \mathbb{R}^{n \times m}$ 
for  $i$  from 1 to  $n$  do
    for  $j$  from 1 to  $m$  do
         $value \leftarrow probText[i][j] - probThird[i][j]$ 
        if  $value \geq threshold$  then
             $output[i][j] \leftarrow value$ 
return  $output$ 

```

dove $probText$ e $probThird$ sono rispettivamente mappe di probabilità di appartenenza alle classi di testo e bounding box / separazione, entrambe generate dopo l'applicazione della funzione di Softmax agli score finali.

4.3.1 BranchFCN

Questa approccio deve il suo nome alla modifica apportata all'FCN fin'ora usata per contrastare un problema sorto in fase di implementazione. Infatti durante un addestramento preliminare si è notato che con l'introduzione della nuova classe delle bounding box le zone con probabilità di essere testo venivano ristrette producendo come effetto indesiderato l'annichilamento di tali predizioni ove il testo fosse di piccole dimensioni.

La soluzione consiste nello sdoppiare la fase di deconvoluzione finale in due rami, uno che generi predizioni per la solo area di testo ed il secondo per le sole bounding box. La scelta di ramificare dopo la fase di classificazione è dettata prettamente dal volere mantenere una certa efficienza, si evita infatti di raddoppiare il numero di pesi totali della rete (presenti perlopiù nella fase finale della VGG) e di addestrare due reti similari separate.

In aggiunta a ciò, l'operazione di **closing** utilizzata nell'algoritmo 2 viene sostituita con un'operazione di **dilation** al fine di recuperare parte dei contorni persi durante l'applicazione dell'algoritmo 3 precedentemente descritto.

Come per il primo training su COCO-Text, anche con questo approccio si è deciso di addestrare la rete per 180mila iterazioni. I risultati visivi sono già da subito promettenti come mostrato nella figura 4.5.



Figura 4.5: Immagine con relativa mappa delle probabilità.

Il grafico della funzione di loss (una media aritmetica fra i due rami dell'architettura) ci mostra come l'apprendimento non venga rallentato dalla presenza di una nuova classe in quanto i valori riflettono quanto già visto con l'addestramento di base su COCO-Text.

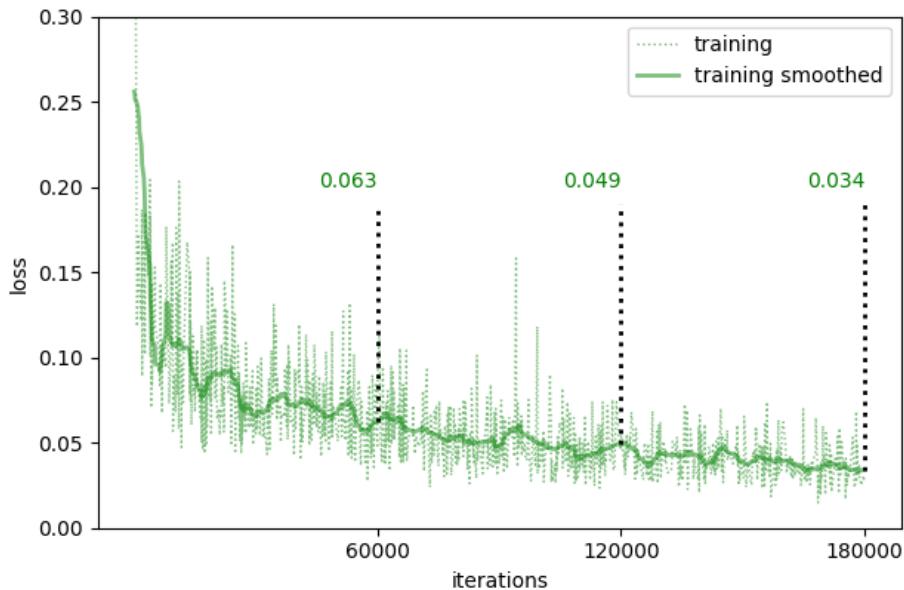


Figura 4.6: Curva di loss su training set.

Nella tabella seguente 4.3 i risultati sul validation set, nella quale si può notare come i miglioramenti generali rispetto all'addestramento base siano notevoli seppur vi sia un lieve decremento nella recall su zone illeggibili.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illeggibili	Totale		
Intero validation set					
180k	35.91	6.98	26.13	21.55	23.62
Almeno un leggibile per immagine					
180k	—	6.74	27.99	35.46	31.29

Valori espressi in percentuale.

Tabella 4.3

4.3.2 Word Division

Come approccio alternativo rimane quello di introduzione di una classe che rappresenti una linea immaginaria di separazione fra due parole abbastanza vicine. A differenza di BranchFCN però qui non è necessario modificare la rete neurale poiché, per come definita, la nuova classe non dovrebbe avere effetti collaterali sulle predizioni delle aree di testo.

V'è però il problema della definizione di un algoritmo per il posizionamento di questa nuova classe nell'immagine di groundtruth. La seguente figura 4.7 dovrebbe chiarire l'idea alla base dello pseudocodice 4, il cui scopo è quello di rilevare i punti di intersezione esterni di due bounding box, compiendo delle espansioni qualora non fossero in contatto diretto.

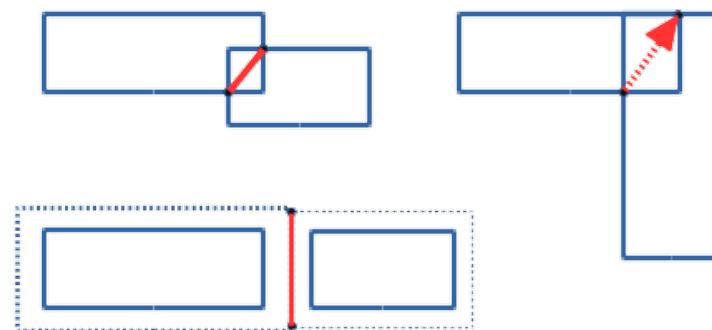


Figura 4.7: I tre esempi principali di estrazione della linea di separazione.

Nella seguente tabella 4.4 i risultati sul validation set, non dissimili a quelli osservati in precedenza con BranchFCN.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illegibili	Totale		
Intero validation set 180k	38.88	8.37	28.70	23.64	25.94
Almeno un leggibile per immagine 180k	—	7.84	30.60	36.74	33.39

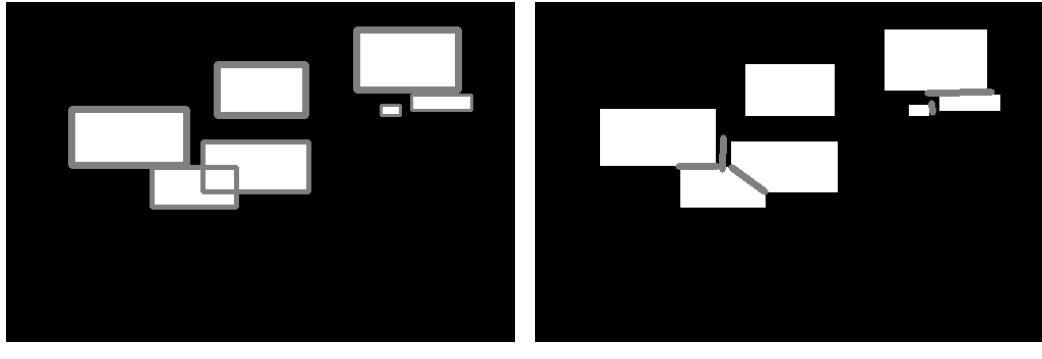
Valori espressi in percentuale.

Tabella 4.4

Algorithm 4 Calcolo e disegno linee di separazione.

```
function DRAWSEPARATIONS(gtImage, bboxes, nExpansion, thickness)
    intersected ← ∅
    for i from 0 to nExpansion do
        for  $b_1 \in bboxes$  do
            for  $b_2 \in bboxes$  do
                if  $(b_1, b_2) \notin intersected$  then
                    line ← GETLINE( $b_1, b_2$ )
                    if line ≠ ∅ then
                        intersected ← intersected ∪ {( $b_1, b_2$ )}
                        DRAWLINE(gtImage, line, thickness)
                for  $b \in bboxes$  do:
                    b ← EXPAND(b)
    return gtImage

function GETLINE( $b_1, b_2$ )
    if  $(b_1 \cap b_2 \neq \emptyset) \wedge (b_1 \not\subseteq b_2) \wedge (b_2 \not\subseteq b_1)$  then
        inter ←  $b_1 \cap b_2$ 
        union ←  $b_1 \cup b_2$ 
        if inter is a Line then
            return inter
        else
            ▷ Vertici su angoli concavi.
            p ← [ $v \mid \forall v \in union.\text{vertices} : \text{ANGLE}(v) > 180^\circ$ ]
            if  $\|p\| = 1$  then
                ▷ Annotationi unite a L ⇒ vertice opposto al concavo
                return  $p \cap inter.\text{vertices}$ 
            else if  $\|p\| = 0$  then
                ▷ Annotationi allineate ⇒ best-fit line
                return FITLINE(inter)
            else if  $\|p\| = 4$  then
                ▷ Annotationi incrociate ⇒ undefined
                return ∅
            else
                ▷  $\|p\| = 2$ 
                return LINE( $p[0], p[1]$ )
    else
        return ∅
```



(a) BranchFCN

(b) Word Division

Figura 4.8: Immagini di groundtruth delle due varianti presentate.

4.4 Finetuning e Data Augmentation

Dopo aver cercato di migliorare l'accuratezza delle predizioni andando a modificare gli output della rete neurale e la rete stessa, per l'ottimizzazione del modello finale rimangono solo approcci quali il *finetuning* e la *data augmentation*, entrambi descritti nello specifico in seguito.

4.4.1 Finetuning

Questa non è la prima volta in cui adottiamo questa tecnica, infatti come già accennato nella sezione 4.1.4, la parte iniziale dell'FCN da noi utilizzata fin'ora è stata sempre inizializzata con pesi derivanti da un precedente addestramento su un task generico di object classification.

Questo potrebbe destare dubbi sull'ottimalità dell'impostazione della rete, ma è stato ormai dimostrato come un approccio del genere sia del tutto sensato a fini di classificazione generale in più ambiti, portando a risparmiare risorse computazionali senza dover partire da un modello ex novo.

L'idea è dunque quella di continuare ad addestrare uno dei modelli da noi precedentemente descritti, nello specifico *Word Division*, utilizzando nuovi dati provenienti da Incidental Scene Text.

Osservando l'andamento della curva di loss riportata nella figura 4.9, i risultati appaiono ottimistici, tant'è che presto i valori vanno ad assestarsi fino a dimezzarsi dopo circa 40mila iterazioni.

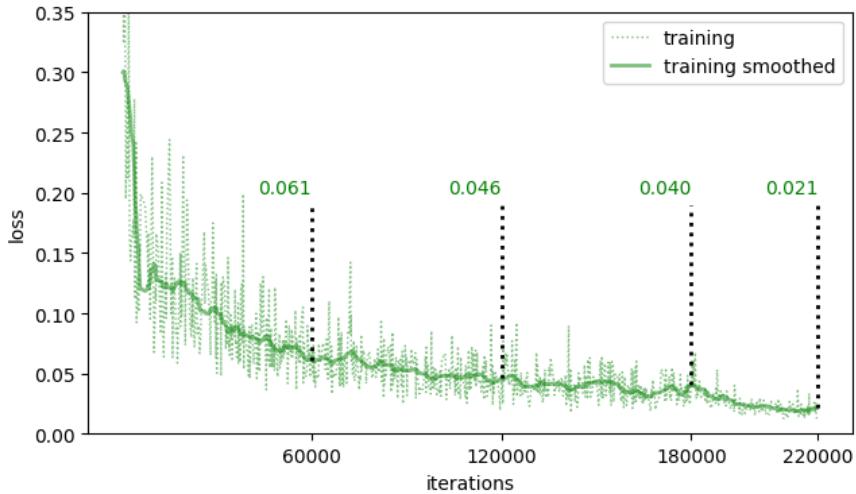


Figura 4.9: Curva di loss su COCO-Text prima e Incidental dopo.

Purtroppo i risultati in validazione, riportati nella tabella 4.5, mostrano una realtà abbastanza diversa, anche se non a noi estranea.

Infatti, seppur la recall abbia un notevole incremento, lo stesso non si può dire della precision che, proprio come successo addestrando la rete da zero su Incidental Scene Text, tende a diminuire drasticamente. Ancora, questo comportamento non ha spiegazioni certe, l'unica ipotesi avanzata rimane quella della già citata differente natura delle annotazioni tra i due dataset.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illegibili	Totale		
Intero validation set					
180k	16.21	4.14	12.28	13.85	13.01
220k	24.99	11.26	20.54	11.95	15.11
Almeno un leggibile per immagine					
180k	—	3.41	12.79	24.11	16.71
220k	—	7.84	21.55	19.94	20.72

Valori espressi in percentuale.

Tabella 4.5

4.4.2 Data Augmentation

Tale approccio consiste nell'aumentare la quantità di dati disponibile da fornire alla rete neurale manipolando quelli già in nostro possesso. Anche questa idea non dovrebbe giungere nuova, infatti già nell'algoritmo 1 di generazione di ritagli randomici viene adottata una tecnica simile, in quanto per qualsiasi immagine non solo vi possono essere tanti ritagli quante le annotazioni, ma possono esservene di molteplici anche per la stessa parola.

Scopo di questo nuovo sviluppo è dunque quello di introdurre nuove forme di *data augmentation*, illustrate nella figura 4.10, nello specifico:

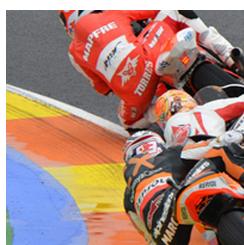
- Scambio dei canali RGB
- Rotazione di un angolo tra $[-\theta, \theta]$

Come primo tentativo vengono applicate entrambe le trasformazioni sempre su COCO-Text, ponendo $\theta = 45^\circ$, utilizzando una FCN base. Già i primi risultati a 60mila iterazioni riportati nella tabella 4.6 evidenziano un comportamento già incontrato, ovvero il calo nella precisione rispetto al metodo di base.

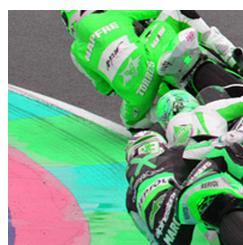
Iterazioni	Recall			Precision	F1-score
	Leggibili	Illeggibili	Totale		
Intero validation set					
60k	15.39	6.66	12.53	7.29	9.22
Almeno un leggibile per immagine					
60k	—	5.70	12.82	14.53	13.62

Valori espressi in percentuale.

Tabella 4.6



(a) Originale



(b) Scambio canali



(c) Rotazione

Figura 4.10

Questi risultati però ci permettono di indicare una possibile causa di questi fenomeni. Gli unici fattori imputabili sono infatti le trasformazioni applicate al dataset, e tra queste solo la rotazione potrebbe portare a somiglianze con Incidental Scene Text, il quale, a differenza di COCO-Text, presenta annotazioni precise sempre ruotate di qualche grado rispetto agli assi verticali e orizzontali.

4.4.3 Confronto tra trasformazioni

Dati i risultati del precedente esperimento si è deciso di optare per questo nuovo tentativo di procedere con due approcci differenti da confrontare.

- Scambio canali RGB
- Scambio canali RGB + Rotazione $\theta = 15^\circ$

La motivazione di questa scelta risiede nella volontà di testare quanto possa influire la rotazione, anche se minima, sulle performance finali. In aggiunta a ciò si passerà ad utilizzare come modello base *Word Division* e l'addestramento partirà in finetuning su quest'ultimo da 120mila iterazioni fino a 180mila.

Confrontando i nuovi risultati nella tabella 4.7 con quelli ottenuti su *Word Division* di base nella tabella 4.4, possiamo notare come le differenze prestazionali siano minime e come la precision tenda sempre a calare (anche se impercettibilmente con $\theta = 15^\circ$) in favore di una migliore recall.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illegibili	Totale		
Intero validation set					
Shuffle	38.34	8.96	28.52	25.40	26.87
Shuffle+15	40.39	7.78	29.43	24.70	26.86
Almeno un leggibile per immagine					
Shuffle	—	8.40	30.30	37.06	33.34
Shuffle+15	—	7.16	31.34	36.60	33.80

Valori espressi in percentuale.

Tabella 4.7

4.4.4 Addestramento finale

Infine, per l'ultimo addestramento si è deciso di utilizzare nuovamente *Word Division* applicando sul dataset lo scambio dei canali RGB e la rotazione di 15°, in quanto questo è l'approccio che fin'ora ha raggiunto risultati mediamente migliori. A differenza di quanto fatto in precedenza però il training ripartirà da zero sempre fino a 180mila iterazioni.

Nella tabella 4.8 riportante i risultati finali è possibile notare come nella localizzazione di zone di aree di testo leggibili questo nuovo metodo si posizioni al di sopra di tutti gli altri, mentre complessivamente raggiunge la terza posizione, superato dai due metodi descritti nella sezione precedente 4.4.3. Le differenze nelle performance risultano comunque non significative.

Iterazioni	Recall			Precision	F1-score
	Leggibili	Illeggibili	Totale		
Intero validation set					
180k	39.58	9.57	29.51	23.66	26.26
Almeno un leggibile per immagine					
180k	—	8.99	31.34	36.84	33.87

Valori espressi in percentuale.

Tabella 4.8

In conclusione è possibile affermare come gli ultimi tentativi di *data augmentation* abbiano portato miglioramenti negligibili in quanto le operazioni applicate sono poco significative rispetto a quanto già fatto nel modello base con l'utilizzo dei ritagli dettati dalle annotazioni.

4.4.5 Tabelle riassuntive

Di seguito riportati i risultati dei vari metodi sviluppati mostrando:

- recall, precision e F1-score totali (tabella 4.9)
- recall, precision, e F1-score solo su annotazioni leggibili (tabella 4.10)
- recall specifica per le varie classi (tabella 4.11)

Dataset	Metodo	Iters	Recall	Precision	F1-score
Incidental Scene Text	FCN	60k	14.54	7.87	10.21
COCO Shuffled 45°	FCN	60k	12.53	7.29	9.22
COCO-Text	FCN	60k	12.28	13.85	13.01
COCO-Text	FCN	180k	17.29	17.72	17.50
COCO + Incidental	FCN	220k	20.54	11.95	15.11
COCO-Text	BranchFCN	180k	26.13	21.55	23.62
COCO-Text	WordDivision	180k	28.70	23.64	25.94
COCO Shuffled	WordDivision	180k	28.52	25.40	26.87
COCO Shuffled 15°	WordDivision	180k	29.43	24.70	26.86
COCO Shuffled 15°	WordDivision	180k	29.51	23.66	26.26

Valori espressi in percentuale.

Tabella 4.9: Risultati sull'intero validation set.

Dataset	Metodo	Iters	Recall	Precision	F1-score
Incidental Scene Text	FCN	60k	15.48	13.95	14.67
COCO Shuffled 45°	FCN	60k	12.82	14.53	13.62
COCO-Text	FCN	60k	12.79	24.11	16.71
COCO-Text	FCN	180k	17.78	28.15	21.79
COCO + Incidental	FCN	220k	21.55	19.94	20.72
COCO-Text	BranchFCN	180k	27.99	35.46	31.29
COCO-Text	WordDivision	180k	30.60	36.74	33.39
COCO Shuffled	WordDivision	180k	30.30	37.06	33.34
COCO Shuffled 15°	WordDivision	180k	31.40	36.60	33.80
COCO Shuffled 15°	WordDivision	180k	31.34	36.84	33.87

Valori espressi in percentuale.

Tabella 4.10: Risultati sulle sole annotazioni leggibili.

Dataset	Metodo	Iters	Recall					
			Leggibili			Illeggibili		
			MP ¹	HW ²	OT ³	MP ¹	HW ²	OT ³
Incidental Scene Text	FCN	60k	19.40	15.51	12.78	19.07	6.13	7.10
COCO Shuffled 45°	FCN	60k	15.46	14.78	13.57	15.39	7.27	9.51
COCO-Text	FCN	60k	16.56	10.78	12.72	16.21	4.57	6.04
COCO-Text	FCN	180k	21.13	16.15	19.39	20.87	10.72	10.17
COCO + Incidental	FCN	220k	25.31	20.54	20.73	24.99	12.29	13.06
COCO-Text	BranchFCN	180k	36.36	30.66	27.61	35.91	7.94	7.48
COCO-Text	WordDivision	180k	39.61	31.10	24.17	38.88	9.37	10.34
COCO Shuffled	WordDivision	180k	39.00	30.88	25.42	38.34	9.72	13.10
COCO Shuffled 15°	WordDivision	180k	41.01	33.33	28.73	40.39	8.59	9.38
COCO Shuffled 15°	WordDivision	180k	40.18	32.84	28.33	39.58	10.55	12.00

Valori espressi in percentuale.

¹ Machine Printed

² Handwritten

³ Others

Tabella 4.11: Valori di recall per le diverse classi.

4.5 Confronto con lo stato dell'arte

In quest'ultima parte verranno mostrati i risultati ottenuti sui test set a disposizione al fine di mettere alla prova i modelli da noi addestrati contro metodi sviluppati da terzi negli ultimi anni.

Dalla nostra parte, i modelli presi in considerazione sono:

- BranchFCN (sezione 4.3.1)
- Word Division con dataset aumentato (sezione 4.4.4)
- BranchFCN con soglia 0.65 e doppia **dilation**
- Word Division con dataset aumentato con soglia 0.6

Gli ultimi due modelli citati non sono stati prima analizzati in quanto sono variazioni nate nel corso di questi ultimi test al fine di massimizzare le prestazioni su *Incidental Scene Text*. In aggiunta a ciò, gli input della rete sono ridimensionati a risoluzioni vicine a 640×480 , media delle immagini di COCO-Text, in quanto alte risoluzioni portano ad una perdita di accuratezza facendo risaltare dettagli fuorvianti.

Come già accennato i test set sono quelli di *Incidental Scene Text* e *Focused Scene Text*, mentre i metodi di terzi riportati saranno quelli dei vincitori delle rispettive edizioni di ICDAR (2015 e 2013 rispettivamente) più i vari contendenti che hanno ottenuti i migliori risultati durante le successive edizioni, seppur non ufficialmente.

4.5.1 Incidental Scene Text

Metodo	Recall	Precision	F1-score
BranchFCN	33.75	44.34	38.33
BranchFCN Tweaked	37.02	54.19	43.99
WordDivision	34.52	38.65	36.47
WordDivision Tweaked	35.29	42.62	38.61
Top-4 2015 [47]			
Stradvision-2	36.74	77.46	49.84
StradVision-1	46.27	53.39	49.57
NJU Text (Version2)	36.25	70.44	47.87
AJOU [48]	46.94	47.26	47.10
Top-3 2017 [47]			
Baidu VIS	83.39	93.62	88.21
SenseTime V3	84.83	90.82	87.73
Dahua-OCR V3	83.44	91.31	87.19

Valori espressi in percentuale.

Tabella 4.12: Risultati sull'intero validation set.

Dai risultati riportati nella tabella 4.12 è possibile notare come nel giro di pochi anni lo stato dell'arte sia avanzato in maniera considerevole, ma è doveroso sottolineare come i nostri risultati ottenuti con strumenti disponibili al pubblico già nel 2015, ci possano posizionare nella classifica di quell'anno in quinta posizione con *BranchFCN Tweaked*, oppure in settima con il modello con parametri di base.

Da notare è l'inversione di performance tra *BranchFCN* e *Word Division* rispetto ai risultati ottenuti su COCO-Text in fase di validazione, soprattutto nelle loro versioni migliorate.

4.5.2 Focused Scene Text

Metodo	Recall	Precision	F1-score
BranchFCN	66.03	59.95	62.84
BranchFCN Tweaked	74.52	66.61	70.34
WordDivision	67.21	57.54	62.01
WordDivision Tweaked	70.96	59.40	64.67
Top-3 2013 [49]			
2R_NUS_FAR	68.95	74.46	71.60
USTB_TexStar	61.46	84.76	71.25
CASIA_NLPR	66.12	74.64	70.12
Top-3 2015 [49]			
VGGMaxNet_cmb [50]	78.08	90.09	83.66
VGGMaxNet_025 [50]	79.82	87.58	83.52
VGGMaxNet_013 [50]	76.53	90.50	82.93
Top-3 2017 [49]			
SenseTime	91.87	95.45	93.62
DLVCtext	88.49	93.35	90.86
MultDet	89.68	91.09	90.38

Valori espressi in percentuale.

Tabella 4.13: Risultati sull'intero validation set.

Similmente a quanto osservato su Incidental Scene Text, nel tempo lo stato dell'arte è notevolmente avanzato e *BranchFCN* è ancora in netto vantaggio rispetto a *Word Division*, anche se solo nella sua versione modificata.

Questa volta però i nostri approcci non raggiungono risultati soddisfacenti, posizionandosi poco dietro i primi classificati nel 2013, ma ben lontani dai modelli concorrenti sviluppati fino al 2015.

Capitolo 5

Conclusione e sviluppi futuri

Questa tesi ha affrontato il problema della localizzazione del testo in immagini digitali con tecniche di *deep learning*.

Come primo passo abbiamo presentato i dataset a nostra disposizione e i metodi di valutazione al fine di analizzare su quanto avremmo dovuto operare e quali obiettivi raggiungere. In seguito abbiamo descritto brevemente le reti neurali e la loro storia col fine di presentare lo strumento principale adottato in questa tesi per l'implementazione, descritta nell'ultima parte, di un nostro metodo per la **text localization**.

5.1 Risultati

L'architettura delle Fully Convolutional Networks è stata cruciale per lo sviluppo del nostro metodo e ci ha permesso di ottenere i risultati aspettati nonostante i dati di partenza scarsi o poco precisi, ma soprattutto differenti da quelli per cui è stata ideata inizialmente.

Questa ha anche mostrato le sue limitazioni sulla precisione della granularità della segmentazione che ci ha obbligato a sviluppare approcci alternativi per superare la difficoltà nel separare parole distinte.

Nonostante questi limiti, il modello architettonico ci ha permesso di compiere con relativa semplicità la fase di localizzazione del testo classificandosi alla pari con metodi concorrenti più sofisticati sviluppati nel periodo in cui questa rete neurale è stata introdotta.

5.2 Sviluppi futuri

I risultati ottenuti però al contempo deludono rispetto ai numeri attualmente raggiunti dallo stato dell'arte. Ciò è da imputare principalmente al fatto che gli strumenti da noi utilizzati, seppur recenti, risultano ormai superati da nuovi sviluppi nella ricerca che continua a fare grandi passi.

Andando a toccare su più frangenti i metodi da noi illustrati, di seguito sono descritti i vari miglioramenti che possono essere adottati, in ordine di complessità.

Modifica Groundtruth

Questo è un obiettivo che in parte abbiamo già sviluppato coi nostri tentativi di utilizzare una nuova classe da predire, ma le idee potrebbero essere molteplici. Fra queste possiamo citare un approccio visto nella già citata **WordFence** [5], ovvero utilizzare una maschera dei pesi per la loss pixel per pixel che faccia pesare di meno eventuali predizioni negative agli estremi delle bounding box.

Deconvoluzione

Come già osservato nella sezione 3.3 la fase di deconvoluzione è divisa in due passi per ritornare alla dimensione originale dell'input, in cui vengono addizionati due dei layer precedenti di pooling. L'idea sarebbe dunque quella di estendere questo procedimento tante volte quante le fasi di pooling in maniera da costruire una fase di deconvoluzione “specchiata” rispetto a quella di convoluzione.

Classificatore

Nella sezione 3.2.3 abbiamo descritto in breve alcuni degli ultimi risultati nella modellazione di reti neurali convoluzionali, fra cui ricade anche la VGG utilizzata alla base delle Fully Convolutional Networks. Come è possibile notare, seppure questa sia il classificatore di riferimento, nulla impedirebbe di rimpiazzarla con uno dei modelli a lei superiori nel task di object detection, portando di conseguenza teoricamente a risultati più precisi.

Estrazione Bounding Box

Infine come tentativo ultimo, anche se non meno importante, rimane quello di sviluppare un nuovo metodo più sofisticato rispetto a quello da noi utilizzato (che ricordiamo essere semplici operazioni morfologiche e di rilevamento di componenti connesse) da mettere in coda alla fase di segmentazione.

L'idea potrebbe essere quella di utilizzare sempre un approccio orientato al machine learning ma questa volta di natura differente, per esempio un task di regressione delle bounding box come già accennato in precedenza nello stato dell'arte [6].

Bibliografia

- [1] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, and Robert Young. Icdar 2003 robust reading competitions. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 682–687. IEEE, 2003.
- [2] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4651–4659, 2015.
- [3] Hojin Cho, Myungchul Sung, and Bongjin Jun. Canny text detector: Fast and robust scene text localization algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3566–3573, 2016.
- [4] Zheng Zhang, Chengquan Zhang, Wei Shen, Cong Yao, Wenyu Liu, and Xiang Bai. Multi-oriented text detection with fully convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4159–4167, 2016.
- [5] Andrei Polzounov, Artsiom Ablavatski, Sergio Escalera, Shijian Lu, and Jianfei Cai. Wordfence: Text detection in natural images with border awareness. *CoRR*, abs/1705.05483, 2017.
- [6] Wenhao He, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Deep direct regression for multi-oriented scene text detection. *arXiv preprint arXiv:1703.08289*, 2017.
- [7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, jun 2010.
- [8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR09*, 2009.

- [9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. Microsoft coco: Common objects in context. In *ECCV*. European Conference on Computer Vision, September 2014.
- [10] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluís Gómez i Bigorda, Sergi Robles Mestre, Joan Mas Romeu, David Fernández Mota, Jon Almazán, and Lluís-Pere de las Heras. Icdar 2013 robust reading competition. In *ICDAR*, 2013.
- [11] Dimosthenis Karatzas, Lluís Gómez i Bigorda, Anguelos Nicolaou, Suman K. Ghosh, Andrew D. Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. Icdar 2015 competition on robust reading. In *ICDAR*, 2015.
- [12] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge J. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *CoRR*, abs/1601.07140, 2016.
- [13] A. Gupta, A. Vedaldi, and A. Zisserman. Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [14] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.
- [15] D.O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 2005.
- [16] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [17] M.L. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1972.
- [18] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.
- [19] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

- [20] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [21] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.
- [22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [23] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.
- [24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- [26] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, 29(32):4790–4797, 1990.
- [27] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
- [28] Dave Steinkraus, I Buck, and PY Simard. Using gpus for machine learning algorithms. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 1115–1120. IEEE, 2005.
- [29] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.
- [30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira,

C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

- [31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [35] Ross Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.
- [36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.
- [37] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.
- [38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [39] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016.
- [40] Guido van Rossum and Jelke de Boer. Interactively testing remote servers using the python programming language. *CWI Quarterly*, 4(4):283–303, 1991.

- [41] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.
- [42] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online].
- [43] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [44] Sean Gillies, Aron Bierbaum, Kai Lautaportti, and O Tonnhofer. Shapely. *GIS-Python Lab*, 2013.
- [45] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [46] MatConvNet pretrained models on the ImageNet ILSVRC classification task. <http://www.vlfeat.org/matconvnet/pretrained/#imagenet-ilsvrc-classification>.
- [47] Incidental Scene Text: Task 1 - Text Localization. <http://rrc.cvc.uab.es/?ch=4&com=evaluation&task=1>.
- [48] Hyung Il Koo and Duck Hoon Kim. Scene text detection via connected component clustering and nontext filtering. *IEEE transactions on image processing*, 22(6):2296–2305, 2013.
- [49] Focused Scene Text: Task 1 - Text Localization (IoU). <http://rrc.cvc.uab.es/?ch=2&com=evaluation&task=1>.
- [50] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.