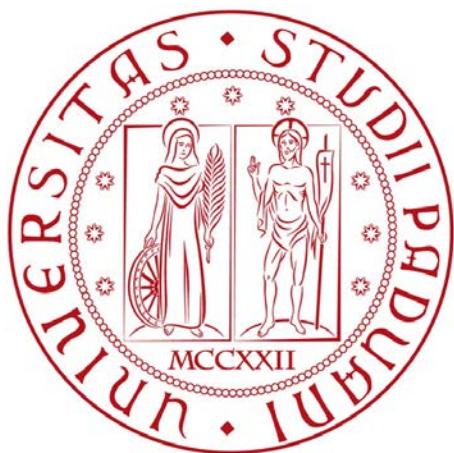


Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA IN INFORMATICA



**Sviluppo di un algoritmo di apprendimento
automatico per l'identificazione dei dati
contenuti in polizze assicurative**

Tesi di laurea triennale

Relatore

Prof.Lamberto Ballan

Laureando

Giovanni Cavallin

Giovanni Cavallin: *Sviluppo di un algoritmo di apprendimento automatico per l'identificazione dei dati contenuti in polizze assicurative* , Tesi di laurea triennale, © Settembre 2018.

Sommario

Il presente documento presenta la relazione finale redatta in seguito al completamento dell'esperienza di stage di due mesi effettuata presso l'azienda RiskApp. È così composto:

1. descrizione del contesto aziendale in cui mi sono inserito;
2. descrizione degli obiettivi di stage;
3. note di teoria sugli argomenti che ho poi affrontato dal punto di vista produttivo;
4. descrizione in dettaglio della progettazione del software prodotto;
5. valutazione retrospettiva dell'esperienza;
6. appendice dove sono presentati i risultati sperimentali prodotti dal software.

Note tecniche

Riguardo la stesura del testo sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzato il colore azzurro;
- i termini in lingua straniera o che fanno parte del gergo tecnico sono evidenziati con il carattere *corsivo*;
- per termini che si riferiscono a prodotti software si utilizzerà invece il colore grigio.

Ringraziamenti

Per questo traguardo desidero ringraziare più di tutti i miei genitori e mia sorella, che non hanno mai smesso di credere in me e nelle mie capacità nonostante le evidenze un tempo contrarie.

Vorrei poi ringraziare Chiara, che mi ha insegnato un nuovo modo di vedere il mondo con la sua delicata presenza.

Ringrazio poi i miei amici: hanno reso questi anni più leggeri e più belli.

Padova, Settembre 2018

Giovanni Cavallin

Indice

1 L’azienda	1
1.1 Contesto organizzativo e produttivo	1
1.2 Tecnologie utilizzate	1
1.3 Processi interni	2
1.4 Tipologia clientela	3
1.5 Innovazione	3
2 Lo stage	5
2.1 L’idea	5
2.2 Aspettative aziendali	5
2.3 Aspettative personali	5
2.4 Obiettivi	6
2.5 Analisi preventiva dei rischi	6
2.6 Pianificazione	7
3 La teoria	9
3.1 Cos’è l’apprendimento automatico	9
3.1.1 Apprendimento supervisionato e non supervisionato	10
3.2 Le reti neurali	10
3.3 Excursus sulle <i>deep network architectures</i>	12
3.3.1 ResNet	13
3.3.2 Inception	14
3.4 Il <i>deep learning</i> applicato all’ <i>object detection</i>	17
3.4.1 Cosa sono il <i>transfer learning</i> e il <i>fine tuning</i>	17
3.4.2 Faster R-CNN	17
3.4.3 SSD	20
3.4.4 Considerazioni finali	21
4 Progettazione	23
4.1 Tecnologie e strumenti utilizzati	23
4.1.1 Codice e versionamento	23
4.1.2 Allenamento di reti neurali e inferenza	24
4.1.3 <i>Editing</i> di immagini	24
4.1.4 Estrazione di informazioni	25
4.2 TableTrainNet: creare il <i>dataset</i>	26
4.2.1 <i>Overview</i> generale	26
4.2.2 Il <i>dataset</i> iniziale	26
4.2.3 Modifica delle immagini per migliorare l’apprendimento	27

4.2.4	Preparazione delle etichette	28
4.2.5	Creazione dei file <i>TFRecord</i>	28
4.2.6	Allenamento della rete neurale personalizzata	28
4.3	IntelligentOCR	33
4.3.1	<i>Overview generale</i>	33
4.3.2	<i>Design pattern</i> utilizzati	33
4.3.3	La <i>pipeline</i>	33
4.3.4	Risultati ottenuti	35
5	Valutazione retrospettiva	37
5.1	Raggiungimento degli obiettivi	37
5.2	Resoconto dell'analisi dei rischi	39
5.3	Possibili miglioramenti futuri	39
5.4	Conoscenze acquisite e valutazione finale	40
A	Appendice	41
A.1	<i>Boxes</i> rilevate nei test	41
A.1.1	Test 0	42
A.1.2	Test 1	49
A.1.3	Test 2	56
A.1.4	Test 3	63
A.1.5	Test 4	70
A.1.6	Test 5	77
A.2	<i>Crop</i> da polizze assicurative	84
A.2.1	Test 0	85
A.2.2	Test 1	87
A.2.3	Test 2	89
A.2.4	Test 3	91
A.2.5	Test 4	93
A.2.6	Test 5	95
	Glossario	97
	Acronimi	101
	Bibliografia	103

Capitolo 1

L’azienda



Figura 1.1: Logo RiskApp

RiskApp è il principale fornitore di tecnologia che possiede una piattaforma di gestione del rischio a tutto tondo per il settore assicurativo. È stata concepita per supportare e migliorare le sottoscrizioni, i reclami, le vendite e le decisioni tecniche riguardanti le coperture assicurative: offre quindi dati e analisi dei rischi, consulenza e sviluppo di software per la valutazione dei rischi aziendali.

RiskApp possiede un algoritmo proprietario che stima il valore del rischio in base a dati raccolti da più fonti ed esprime le perdite economiche che possono derivare da tali rischi, confrontando i risultati con le migliori pratiche del settore.

1.1 Contesto organizzativo e produttivo

Nell’azienda lavorano come lavoratori autonomi il fondatore, Federico Carturan, il co-fondatore e tecnico informatico Pierpaolo Toniolo e Luca Bizzaro, ingegnere civile. L’ufficio è ubicato a Conselve, all’interno del quale ognuno lavora con la propria macchina. Il *core business* dell’azienda è la fornitura di servizi ad agenti assicurativi, fornendo costantemente nuove funzionalità per la loro già esistente piattaforma RiskApp tramite sviluppo di migliori interfacce software e il miglioramento dell’algoritmo proprietario.

1.2 Tecnologie utilizzate

Il sistema è composto da un *frontend* e un *backend*. Il codice riguardante il *backend* è sviluppato in linguaggio Python 3 e gestito dal web framework Django[1].

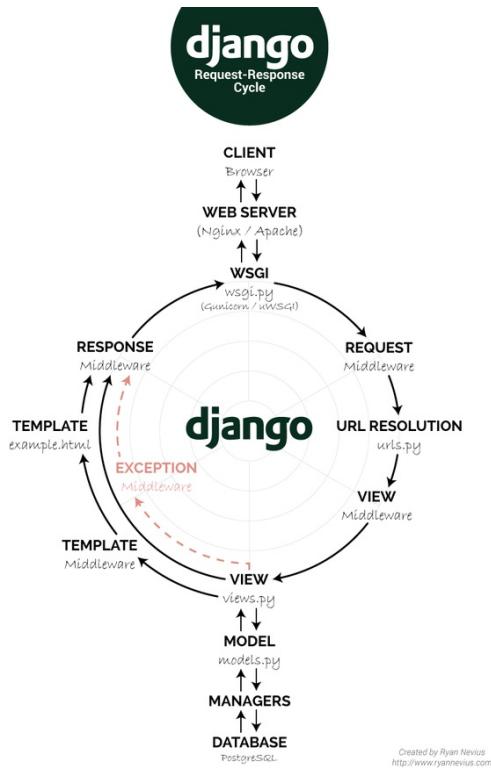


Figura 1.2: Funzionamento Django

Il codice riguardante il *frontend* è sviluppato in linguaggio JavaScript, facendo uso del framework React[2] utilizzando come *state container* Redux[3]. La comunicazione tra *frontend* e *backend* è costituita da chiamate REST attraverso l'utilizzo di Django REST framework[4]. Per Python viene utilizzato PyCharm[5], un IDE prodotto da JetBrains® che, oltre ad offrire un assistente "intelligente" per la scrittura di codice e *safe refactoring*, include anche degli strumenti di sviluppo quali *debugging* e *deploying* su GitHub[6]. Per quanto riguarda invece JavaScript, viene utilizzato IntelliJ IDEA[7], che offre le stesse funzionalità di PyCharm.

1.3 Processi interni

Il codice è organizzato all'interno di due repository GitHub[6] ospitate nel profilo dell'azienda. In particolare, c'è una *repository* per il versionamento del *frontend* e una per quello del *backend*. Lo sviluppo del codice avviene in rami separati per ogni sviluppatore; al compimento delle *features* per una *milestone* viene fatta una *pull request* nel ramo *development*. Quindi, dopo un periodo di test nel server di testing, viene fatta una *pull request* nel ramo *master* e quindi il *deploy* nell'ambiente di produzione.

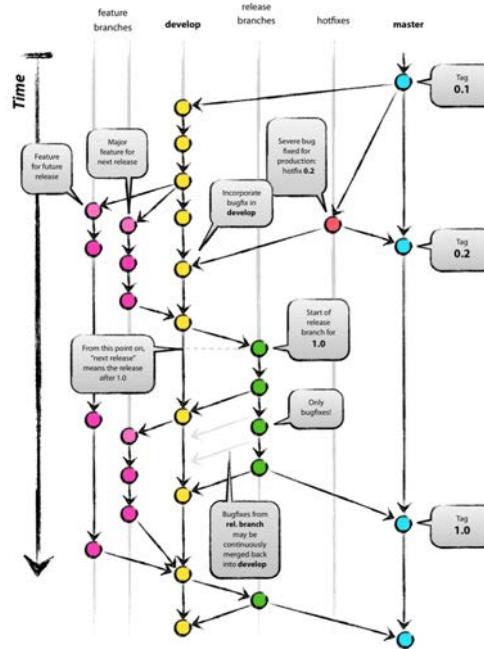


Figura 1.3: Distribuzione del lavoro su rami git

Il processo di verifica del codice è automatizzato da script interni che consentono una *continuous integration*. Tali test vengono lanciati ogni volta che avviene un *push* attraverso l'ausilio di CircleCI[8], servizio che è integrato con GitHub e che ad ogni *push* in automatico testa la *build* in una macchina virtuale creata appositamente per i test.

Invece, la gestione delle *milestones* viene fatta attraverso i *projects* di GitHub, in cui vengono fatti convogliare i rispettivi *task* necessari per lo sviluppo delle *features* selezionate. Ogni compito rappresenta un'azione atomica necessaria per l'implementazione della *feature* ed è rappresentata da una *issue* in GitHub. Ogni *issue* viene quindi etichettata per tenere traccia del tipo di azione o priorità che identifica.

1.4 Tipologia clientela

La piattaforma viene principalmente utilizzata da professionisti del rischio, brokers e agenti assicurativi, che la utilizzano per reperire informazioni dettagliate sui i rischi assicurabili e per ricevere un preventivo in maniera istantanea di eventuali polizze.

1.5 Innovazione

RiskApp è una startup concentrata nell'innovazione del settore assicurativo, l'*Insurtech*. Il tutto è nato dalla selezione in Unipol Ideas, il percorso di Open Innovation del Gruppo Unipol: un canale di dialogo e contaminazione con chi fa innovazione in ambiti collegati al core business aziendale. Unipol è tra i soci fondatori della Fondazione ItaliaCamp. Riskapp si è presentata in Expo Milano 2015 al Vivaio delle Idee, lo spazio dedicato all'innovazione ideato e gestito da Padiglione Italia, Ministero delle Politiche

Agricole e Forestali e Fondazione ItaliaCamp[9].

Nel 2016 Unipol è diventata il primo cliente, inoltre sono state fatti altri programmi di accelerazione in Europa, tra i quali Deloitte Digital Disruptors[10] a Lisbona, MundiLab[11] a Madrid, Fintech Innovation Lab[12] a Londra che hanno portato alla collaborazione con Società di consulenza quali Deloitte[13] e Accenture[14], riassicuratori Munich Re[15] e Swiss Re[16], e a nuovi clienti in Europa, come Fidelidade[17] e P&V[18].

Capitolo 2

Lo stage

2.1 L'idea

Lo stage è stato proposto durante l'incontro STAGE-IT[19] tra le aziende e gli studenti di informatica e di ingegneria informatica. L'idea all'interno del quale si innesta lo stage proposto era la possibilità, da parte di un'agente assicurativo, di caricare una polizza assicurativa o il libretto e di poter ricavare, automaticamente, tutti i dati salienti in esso contenuti in maniera automatica. Quindi: ubicazioni, dimensioni, premi assicurativi, coperture, eccezioni, eccetera.

Dall'idea si è poi passati alla pratica: il primo passaggio sarebbe stato quello di creare un lettore **OCR** di PDF per così dire "intelligente", che quindi fosse in grado di comprendere dove fossero posizionate le tabelle e il testo per poter poi analizzare le due tipologie di strutture dati in maniera differente e più efficace.

2.2 Aspettative aziendali

L'azienda RiskApp sperava, grazie allo stage, di poter iniziare a mettere "le mani in pasta" in un ambiente completamente non sondato del suo business. La possibilità di avere uno strumento del genere nel proprio portale le permetterebbe infatti di ampliare il *set* di strumenti già in loro possesso, facilitando la navigazione e il reperimento delle informazioni da parte dell'utilizzatore. Inoltre ha un importante ruolo nella generazione di un database ordinato, all'interno del quale inserire tutte quelle informazioni specifiche che, se estrapolate a mano, avrebbero un costo proibitivo.

2.3 Aspettative personali

Ho affrontato l'evento STAGE-IT con il desiderio di poter trovare un'azienda che mi proponesse uno stage riguardante l'intelligenza artificiale. Fino ad allora mi ero solamente informato in maniera generica sull'argomento: volevo finalmente imparare a utilizzare gli algoritmi di *machine learning* e *deep learning* e capire come declinarli nell'ambito aziendale. Invero, tra le oltre 80 aziende e molte più offerte di stage, solo due o tre offrivano questa possibilità e RiskApp mi ha convinto per il fatto che è una realtà giovane - è ancora registrata come StartUp. Ero infatti anche molto incuriosito dall'idea di azienda fortemente dinamica.

2.4 Obiettivi

Gli obiettivi stilati ad inizio stage erano molto audaci e, sebbene questo abbia portato la ridefinizione del piano di lavoro in corso d'opera, ha aiutato l'azienda a capire la complessità dell'argomento e spronato me a velocizzare il più possibile il processo conoscitivo e produttivo.

Dagli obiettivi presentati dall'azienda ho estrapolato dei requisiti, che seguiranno le seguenti notazioni:

- **O** per gli requisiti obbligatori;
- **D** per gli requisiti desiderabili, quindi non vincolanti ma dal valore aggiuntivo;
- **F** per gli requisiti facoltativi

Nella tabella ogni notazione sarà seguito da un numero, per poter identificare ogni obiettivo univocamente.

ID	Descrizione
Obbligatori	
O01	Creazione di un algoritmo che faciliti la creazione di un dataset per allenare una rete neurale
O02	Creazione di una struttura di testing per la valutazione visiva dei risultati ottenuti
O03	Creazione di un algoritmo di estrazione di informazioni mirate da un libretto assicurativo
O04	Gestione di tutti gli algoritmi il più automatica possibile, possibilmente tramite l'uso di costanti
O05	Utilizzo del linguaggio Python versione 3
Desiderabili	
D01	Creazione di un batch multithreading per l'estrazione parallela di informazioni da più polizze
Facoltativi	
F01	Integrazione del sistema nell'infrastruttura dell'azienda

Tabella 2.1: Tabella degli obiettivi

2.5 Analisi preventiva dei rischi

Durante la fase iniziale ho individuato alcuni possibili rischi che questo progetto mi avrebbe potuto portare ad affrontare, consentendomi di elaborare una strategia di conseguenza:

Descrizione	Piano di emergenza	Rischio
Non conoscenza del linguaggio: prima di iniziare lo stage non avevo mai programmato in Python.	Escogitare dei meccanismi di apprendimento rapido del linguaggio, tramite corsi online e provando fin da subito a scrivere del codice per conto proprio. Chiedere agli altri stagisti qualche domanda più tecnica che mi potesse risparmiare del tempo per la ricerca.	Occorrenza: Alta Pericolosità: Media
Non conoscenza della materia: Prima dell'inizio dello stage la conoscenza teorica in mio possesso di <i>machine learning</i> , <i>deep learning</i> e di reti neurali era generale e per niente applicata.	Frequentare il prima possibile dei corsi specializzanti.	Occorrenza: Alta Pericolosità: Alta
Non conoscenza degli strumenti: Per il <i>machine learning</i> e il <i>deep learning</i> esistono tutta una serie di strumenti già pronti all'uso che risparmiano molto tempo in fase di progettazione e codifica.	Mettere mano il prima possibile a questa strumentazione, così da entrare in confidenza fin da subito con le tecnologie e familiarizzare con le personalizzazioni che si possono adoperare.	Occorrenza: Alta Pericolosità: Media
Esami durante il lavoro: Lo stage è iniziato senza avere la certezza di aver superato un esame, che avrei dovuto quindi sostenere durante il periodo di lavoro.	Pianificare il lavoro in maniera tale da lasciare fasi meno impegnative della programmazione - come la fase di test - nel periodo subito precedente l'esame. Così la sera sarei riuscito a studiare.	Occorrenza: Alta Pericolosità: Bassa

Tabella 2.2: Tabella dell'analisi dei rischi

2.6 Pianificazione

Dopo un'analisi approfondita degli obiettivi richiesti dall'azienda, dopo essermi reso conto dei rischi e in accordo col tutor aziendale, ho capito che la pianificazione avrebbe dovuto comprendere un numero di ore molto alto per la formazione prima di poter iniziare a sviluppare l'applicazione vera e propria.

Ho intuito immediatamente poi che la creazione di una "catena di montaggio" sarebbe stata da preferire rispetto ad un approccio monolitico. Questo mi avrebbe permesso di capire più rapidamente il funzionamento di ogni singola parte della *pipeline* (di cui si discuterà in sezione 4), potendola quindi testare e migliorare man mano grazie ai risultati ottenuti da altre parti della catena. L'effetto collaterale di questo approccio è che il risultato finale, dato dall'assemblaggio dei vari pezzi, sarebbe arrivato più tardi di quanto un approccio monolitico avrebbe potuto offrire.

Ore	Descrizione dell'attività
24	Raccolta informazioni e definizioni dei requisiti del progetto;
100	Formazione su: <ul style="list-style-type: none"> • python; • openCV; • algoritmi di <i>machine learning</i> e <i>deep learning</i> e loro ottimizzazione; • basi di Tensorflow; • basi di Keras.
40	Creazione della struttura per l'allenamento di reti neurali;
80	Creazione di un algoritmo per l'estrazione di dati dai documenti;
24	Creazione di test automatici per la valutazione visiva di una rete neurale;
18	(Desiderabile) Creazione di un batch multithreading per un'analisi estensiva su più polizze;
18	(Facoltativo) Integrazione nell'infrastruttura aziendale.
Totale: 304 ore	

Tabella 2.3: Tabella della suddivisione delle ore

Capitolo 3

La teoria

Per comprendere al meglio le tecnologie e i concetti che avrei dovuto implementare in questo stage ho deciso di avere un approccio il più sistematico possibile alla materia. Essendo la teoria implicata molto ampia e con un approccio matematico non banale, ho deciso inizialmente di leggermi molti articoli, per lo più tratti da [medium.com](#)[20], ottimo sito di divulgazione generale. Dopodiché ho frequentato alcuni corsi su Coursera[21], famoso sito di corsi online.

In particolare, ho seguito buona parte del corso sul *machine learning*[22] tenuto da Andrew Ng[23] per avere un'infarinatura generale di cosa vuol dire questo approccio matematico. In seguito per avere un'idea più completa riguardo le *deep networks* mi sono affidato a dei corsi, sempre su Coursera[21] e tenuti dallo stesso docente, che fanno parte di una specializzazione[24]. In particolare, ho completato - con certificato - i seguenti corsi:

- Neural Networks and Deep Learning[25];
- Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization[26];
- Structuring Machine Learning Projects[27];

Non sono invece riuscito a concludere gli esercizi finali del quarto capitolo: Convolutional Neural Networks[28].

Tutta questa preparazione mi è stata di fondamentale aiuto per comprendere le tecnologie che sarei andato ad utilizzare di lì a breve, nonché mi ha dato la possibilità di ampliare enormemente la mia conoscenza al riguardo. Nei capitoli seguenti descriverò in maniera più intuitiva che matematica le tecnologie che ho esplorato.

3.1 Cos'è l'apprendimento automatico

L'apprendimento automatico - in inglese *machine learning* - si basa sull'idea che un algoritmo sia capace di dire qualcosa di interessante riguardo un insieme di dati senza la necessità, da parte di un utente, di dover scrivere alcuna riga di codice specifico sul problema.[29] L'algoritmo infatti, sulla base di questi dati, è capace di sviluppare una propria logica. Prendiamo ad esempio due task:

- riconoscere dei numeri scritti a mano;

- riconoscere una mail di spam da una legale.

Possiamo sfruttare lo stesso algoritmo di apprendimento automatico per generare due logiche differenti solamente a partire da due *set* di dati diversi.

3.1.1 Apprendimento supervisionato e non supervisionato

Esistono due principali categorie di apprendimento automatico: supervisionato e non supervisionato.[30]

Apprendimento supervisionato

Il *supervised learning* è principalmente usato per la classificazione, dove vogliamo ottenere delle etichette a partire da dati in ingresso, oppure per la regressione, dove l'input è mappato in un output continuo. In entrambi gli aspetti, l'obiettivo è quello di dedurre una particolare relazione o struttura nei dati di input che ci permetta di produrre un output corretto.

Apprendimento non supervisionato

Con l'*unsupervised learning* si vuole trarre una struttura a partire da dati in ingresso senza fornire esplicitamente un'etichetta. Si usa principalmente per il raggruppamento e le stime di densità di un insieme di dati in input.

3.2 Le reti neurali

Le reti neurali sono una classe di modelli di apprendimento automatico che hanno rivoluzionato il mondo del *machine learning*. Lo sviluppo delle *neural networks* è stata la chiave per insegnare ad una macchina a *capire* come un umano.

Cos'è una rete neurale

La rete neurale si può pensare come una funzione composta. Essa accetta alcuni tipi di input e genera degli output. I componenti della rete neurale sono i **neuroni** (anche detti *Perceptrons* oppure unità), che a loro volta sono una funzione che utilizza **pesi** (o anche detti connessioni) e **biases**. Viene poi interpellata una *funzione di attivazione* non lineare per restringere l'output ad un certo intervallo. Se ne immaginassimo una di combinazione lineare:

$$f(x_1, x_2) = W_1 * x_1 + W_2 * x_2 + b$$

allora il neurone sarebbe la funzione, W_1 e W_2 sarebbero i pesi e b il *bias*. I neuroni sono organizzati in strati - altresì chiamati *layers* - e il numero di neuroni all'interno di ogni strato è a carico del creatore. Tuttavia, troppo strati per un compito semplice porta ad aumentare la complessità della rete senza motivo e a diminuirne l'accuratezza. È ovviamente vero anche il contrario. Per ogni strato, l'output di un neurone viene utilizzato come input per un altro neurone in un altro strato.

Ogni rete ha due tipologie di strati: lo strato per l'input (che non ha alcun tipo di computazione al suo interno) e quello per l'output. Tutti gli altri *layers* che sono tra questi due sono chiamati nascosti. Nell'immagine 3.1 si può notare una rete neurale composta da uno strato di input con otto unità, uno di output con 4, tre nascosti con 9 unità ciascuno. Una rete neurale con più di due strati nascosti è definita profonda.

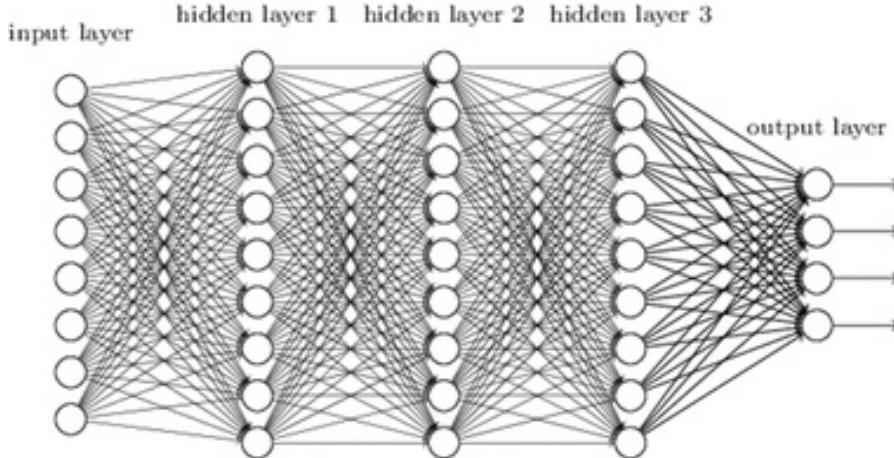


Figura 3.1: Una rete neurale profonda

Come la rete impara

La rete impara attraverso l'analisi della sua *loss function*. La *loss function* è una funzione che ci dice quanto la nostra rete è buona per un determinato compito. Per capirlo intuitivamente, essa prende ogni dato in input, lo computa attraverso la rete neurale, lo si sottrae al numero che ci aspettiamo sia il risultato e ne prendiamo il quadrato. Dati quindi y il numero che ci aspettiamo, \hat{y} il numero che otteniamo dalla computazione della rete, i l'indice dell'esempio che stiamo prendendo in considerazione:

$$L(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

otteniamo la "distanza" dal risultato che vorremmo ottenere. Imparare diventa quindi un problema di minimo, ovvero vogliamo trovare un risultato che minimizzi la *loss function*. A questo scopo si utilizzano la discesa del gradiente e la *backward propagation*

Discesa del gradiente

Il metodo di discesa del gradiente è una tecnica che consente di determinare i punti di massimo e di minimo di una funzione a più variabili.

Pensando ad una funzione $f(x)$ con $x \in \Re^n$, la direzione di massima discesa (ovvero la direzione verso la quale la funzione "scende" più rapidamente) in un punto \bar{x} è dato dall'opposto del suo gradiente in quel punto, ovvero: $p_k := -\nabla f(\bar{x})$. Questa scelta garantisce che la soluzione tenda ad un punto di minimo di f . Il metodo del gradiente prevede dunque di partire da una soluzione iniziale x_0 scelta arbitrariamente e di procedere in maniera iterativa aggiornandola come:

$$x_{k+1} = x_k + \alpha_k p_k$$

dove $\alpha_k \in \Re^+$ corrisponde alla lunghezza del passo di discesa, la cui scelta diventa quindi cruciale nel determinare la velocità con cui l'algoritmo converge alla soluzione richiesta.

Backward propagation

La propagazione all'indietro viene utilizzata dagli algoritmi di discesa del gradiente per aggiustare i pesi dei neuroni calcolando il gradiente della *loss function*. In pratica, l'algoritmo di ottimizzazione di una rete neurale ripete un ciclo costituito da due fasi: propagazione e aggiornamento dei pesi. Quando un vettore in input viene dato in pasto alla rete, viene propagato "in avanti" attraverso la rete, strato su strato, fino a che raggiunge lo strato di uscita. A questo punto l'output viene paragonato col risultato aspettato usando una *loss function*. A questo punto, l'errore che è risultato viene calcolato per ognuno dei neuroni dello strato di output, dopodiché viene propagato dallo strato di uscita all'indietro nella rete, fino ad associare a tutti i neuroni un valore d'errore che riflette il loro contributo all'output originale.

La *backpropagation* quindi usa questi valori per calcolare il gradiente della *loss function*. Nella seconda fase, questo gradiente viene dato in pasto al metodo di ottimizzazione, che lo usa per aggiornare i pesi, per cercare di minimizzare la funzione di perdita.

Per il mio studio mi sono fin da subito concentrato sulle *deep neural network*, ovvero reti neurali che sono composte da molti strati che comunicano fra loro. Nelle prossime sezioni mostrerò alcuni modelli ben definiti e le architetture che poi ho usato nel mio progetto.

3.3 Excursus sulle *deep network architectures*

Quasi tutto il progresso sull'apprendimento profondo degli ultimi anni per la branca del *computer vision* può essere riassunto in poche architetture di reti neurali. Tralasciando in questa sede tutta la matematica e il codice che vi sta dietro, vorrei introdurre il lettore a comprendere come queste reti funzionano e perché.

Prendiamo ad esempio la libreria Keras[31]: questa libreria, capace di lavorare come *wrapper* attorno a Tensorflow[32], CNTK[33] e Theano[34] offre sei modelli già allenati, *built-in* al suo interno:

- **VGG16**
- **VGG19**
- **MobileNet**
- **ResNet50**
- **Inception v3**
- **Xception**

Le reti VGG seguono l'archetipo - ora in parte superato - delle *convolutionary nets*: sono costituite infatti da una serie di strati convoluzionali, di *max-pooling* e con uno strato *fully-connected* per classificazione nel fondo. Le MobileNet sono una versione ottimizzata per le applicazioni mobile della Xception. Le ultime tre, invece, ridefiniscono veramente il modo in cui pensiamo alle reti neurali.

3.3.1 ResNet

La domanda chiave che gli sviluppatori si sono fatti pensando al modello di questa rete è stata: *Perché ogni rete profonda ha una prestazione peggiore man mano che si aggiungono strati?*[35] Infatti ci si può aspettare che, dati n strati, una rete con $n+1$ abbia la stessa identica accuratezza di quella a n . Tuttavia la pratica smentisce questa osservazione, mettendo in risalto l'opposto.

L'ipotesi che gli autori di ResNet hanno fatto è stata che *mappings* diretti sono difficili da imparare. Così hanno proposto una modifica: al posto di cercare di stimare una funzione G che dato un x restituisca $G(x)$, è meglio apprendere la *differenza* tra i due - anche chiamato residuo, da cui il nome. Di conseguenza, per calcolare $G(x)$ a partire da x basta aggiungere il suo residuo.

Quindi, detto:

$$F(x) = G(x) - x$$

il nostro residuo, al posto di apprendere $G(x)$ direttamente, la rete cercherà di imparare

$$F(x) + x$$

. Questo ci dà l'opportunità di introdurre il *ResNet block* (figura 3.2): ogni blocco

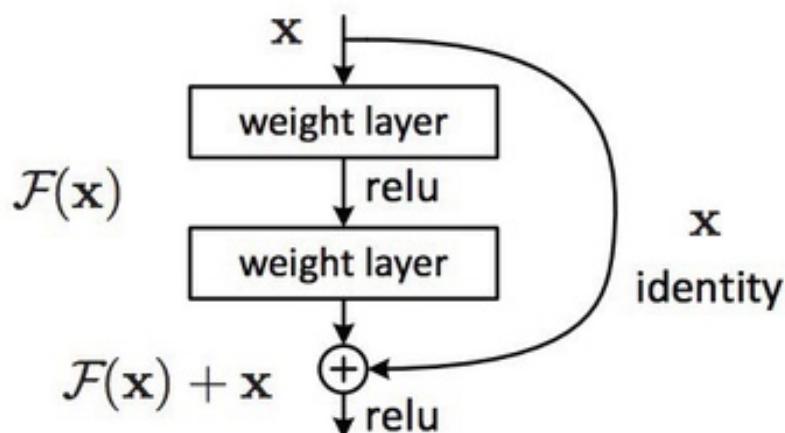


Figura 3.2: Blocco ResNet

è costituito da una serie di strati e da una "scorciatoia", che aggiunge l'input del blocco stesso al suo output. L'operazione di aggiunta è fatta per ogni elemento, e se input e output sono di dimensioni diverse viene eseguito un *zero-padding* oppure una proiezione (via *convoluzione* 1x1) per portare l'input alla stessa dimensione dell'output. Intuitivamente è molto più semplice imparare mandando $F(x)$ a 0 lasciando l'output a x piuttosto di cercare di imparare una trasformazione identità da zero. In altre parole, ResNet dà uno strato di riferimento - x , per l'appunto - da cui iniziare ad imparare.

Quest'idea funziona molto bene nella pratica. Mentre i modelli precedenti soffrivano del cosiddetto *vanishing gradient*, qui invece si possono sfruttare delle scorciatoie verso gli strati precedenti: reti da 50, 100 o addirittura 1000 strati di profondità hanno ancora delle ottime performance.

3.3.2 Inception

La domanda chiave che invece gli sviluppatori della famiglia delle Inceptions si sono fatti è stata: *come possiamo scalare efficientemente le reti neurali senza aumentare il costo computazionale?*[35] Il paper originale[36] si concentra sulla definizione di un "Inception module", che ha come base due concetti fondamentali: le operazioni sugli strati e la possibile riduzione della dimensione di ognuno. Ma andiamo con ordine:

Più operazioni per strato

In una *conv net* tradizionale ogni livello estrae nuove informazioni dallo strato precedente per trasformare i dati in ingresso in una rappresentazione più fruibile per la rete. Tuttavia, ogni tipologia di livello estrae informazioni diverse: l'output di una convoluzione di dimensione 5x5 è diverso da quello di uno a 3x3, ed è diverso anche da uno dove si opera un *max-pooling*. Come possiamo sapere se una certa trasformazione in un certo strato sta estraendo informazioni utili?

L'idea è stata di lasciare la rete decidere quali scegliere: il modulo Inception esegue multiple e diverse trasformazioni sullo stesso input in maniera parallela, concatenando poi i risultati in un output singolo. In pratica, come mostrato in figura 3.3, per ogni strato Inception esegue una convoluzione 5x5, 3x3 e un *max-pooling*. Lo strato successivo si occupa quindi di decidere se e come usare ogni informazione così tracciata.

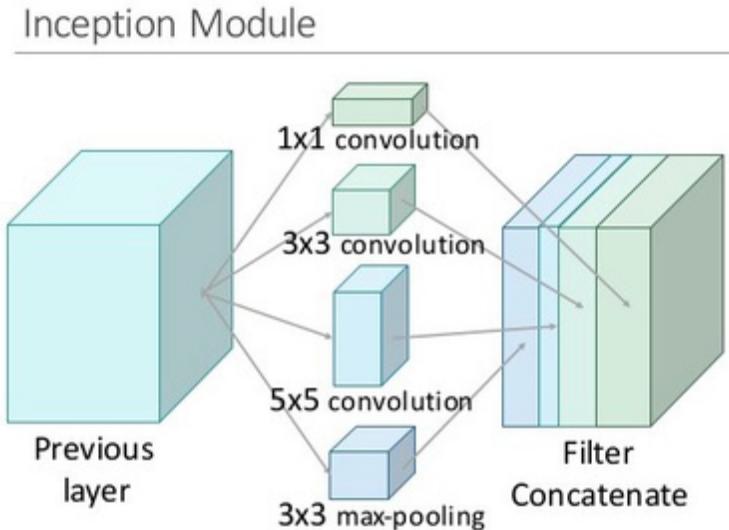


Figura 3.3: Modulo Inception

Riduzione dimensionale

Facendo così tuttavia si introduce un problema: si è ingrandita drasticamente la computazione. Per ogni filtro aggiunto bisogna fare la convoluzione di tutti gli strati di input precedenti per calcolare un singolo output. Questo porta ad un aumento quadratico o addirittura maggiore del numero dei filtri per strato. Ci si rende facilmente

conto che questo non è sostenibile.

Ecco quindi la soluzione: gli autori hanno pensato ad una convoluzione 1x1 per "filtrare" la profondità degli output. Se si considera una convoluzione 1x1 su un singolo strato si ha una valutazione di un valore alla volta. Tuttavia, vista su più canali, essa può estrarre informazioni "spaziali" e comprimerli ad una dimensione inferiore. Facciamo un esempio: usando 20 filtri 1x1, un input di dimensione 64x64x100 può essere compresso a 64x64x20. Così facendo gli autori del modello sono stati capaci di allocare più trasformazioni per strato in parallelo, avendo una rete simultaneamente ampia (per via del parallelismo) e profonda (grazie al grande numero di strati).

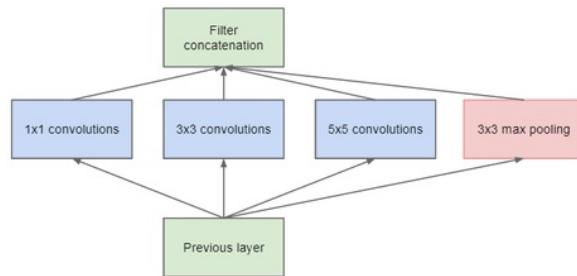


Figura 3.4: Concezione iniziale del modulo Inception

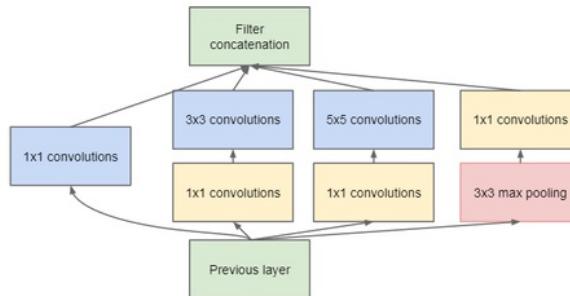


Figura 3.5: Modulo Inception con riduzione

Inception ha dimostrato la potenza delle architetture composte da "reti dentro a reti", aggiungendo un altro gradino importante alla potenza rappresentale delle reti neurali.



Figura 3.6: Meme letteralmente citato dall'articolo originale su Inception

Xception

Xception sta per *extreme inception* e porta il concetto del modulo Inception ad un estremo. L'ipotesi da cui si sviluppa questa rete è: *la correlazione tra i canali e quella spaziale sullo stesso frame è sufficientemente bassa da poter preferire di non mapparli assieme*. Cosa vuol dire? Nelle tradizionali reti a convoluzione, gli strati cercano correlazioni sia in profondità che in larghezza (ovvero sullo stesso frame). Con Inception si ha avuta una prima separazione: usando una convoluzione 1x1 si può mappare l'input originale in vari input separati, sui quali poi poter eseguire diversi tipi di trasformazione. Xception porta il concetto un po' oltre: al posto di partizionare l'input in tanti piccoli pezzi, questa partiziona ogni canale separatamente, quindi esegue una convoluzione 1x1 in profondità per catturare correlazioni tra i canali.

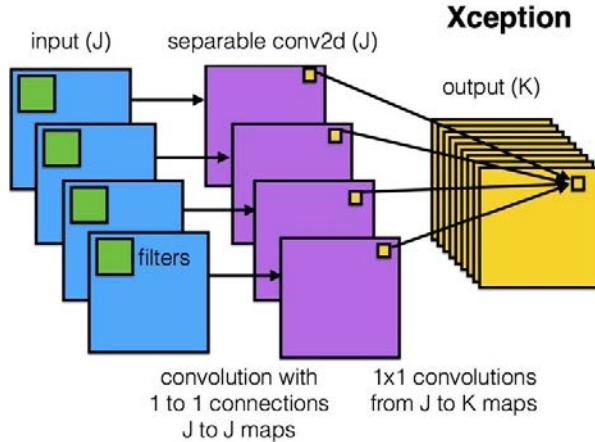


Figura 3.7: Modulo Xception

Possiamo immaginare questa operazione considerando la ricerca di correlazione tra spazi bidimensionali e poi su unidimensionali. Il che è intuitivamente più semplice che cercarle in uno spazio direttamente tridimensionale.

Questo modello è molto recente - è uscito nell'aprile del 2017 - e già ha risultati promettenti, soprattutto per quanto riguarda l'efficienza computazionale con un più alto numero di [classe](#) da analizzare.

3.4 Il *deep learning* applicato all'*object detection*

Con l'aumento delle auto a guida autonoma, dei sistemi di videosorveglianza *smart*, del riconoscimento facciale e di molti altri campi di utilizzo, la richiesta di sistemi di *detection* è andata via via sempre aumentando. Tuttavia, questi sistemi non solo richiedono di classificare ogni oggetto in un'immagine, ma anche di trovarlo, disegnandogli un rettangolo attorno. Come si può immaginare, questo compito è nettamente più complesso della semplice *image classification*. Fortunatamente i migliori approcci usati oggi giorno provengono principalmente da estensioni dei modelli già esistenti e derivati dalla classificazione.

Circa un anno fa Google ha rilasciato una nuova [API](#) di *object detection* e, assieme ad essa, ha fornito anche una serie di architetture - con i relativi pesi già allenati - per alcuni specifici modelli:

- Single Shot Multibox Detector (SSD) con le MobileNets
- SSD con Inception V2
- Region-Based Fully Convolutional Networks (R-FCN) con Resnet 101
- Faster R-CNN con Resnet 101
- Faster R-CNN con Inception Resnet v2

Dopo aver intrapreso uno studio preliminare riguardo le varie architetture, ho preferito concentrarmi su due in particolare: la Faster R-CNN con Inception Resnet V2 e la SSD con Inception V2. Queste infatti sono disponibili all'interno dell'API di *object detection*[\[37\]](#) di Tensorflow[\[32\]](#) e hanno reso il mio approccio alla materia molto meno drastico.

3.4.1 Cosa sono il *transfer learning* e il *fine tuning*

Per usufruire di queste reti ho dovuto utilizzare il *transfer learning*. Molte poche persone allenano un'intera *conv net* da zero con inizializzazione casuale, poiché è raro avere a disposizione un *dataset* così ampio. Piuttosto, si tende ad utilizzare una rete già allenata. quindi usare i pesi - ovvero le informazioni che già ha imparato da un particolare insieme di dati - di questa come di partenza per la propria.

In particolare, per questo progetto ho utilizzato il *fine tuning*: Tensorflow mette a disposizione i pesi "congelati" di tutti gli strati eccetto del penultimo. Così facendo, possiamo allenare la nostra rete ad imparare la rappresentazione del solo ultimo strato, utilizzando il *dataset* personale. Questo è ovviamente possibile solamente se c'è una certa continuità dell'obiettivo della rete: sfruttare il *fine tuning* per imparare una voce utilizzando come base di partenza il grafo di una *object detection* è inutile e controproducente.

3.4.2 Faster R-CNN

La Faster R-CNN è l'architettura di riferimento nell'apprendimento profondo, in particolare per quanto riguarda l'*object detection* ed è di riferimento per altre architetture, in particolare anche per la SSD. Tuttavia, non la si può realmente capire se prima non si dà una rapida occhiata alle sue predecessore, la R-CNN e la Fast R-CNN.

R-CNN

La R-CNN, ovvero la *Region-based Convolutional Neural Network* consiste di tre passaggi:

1. scansiona l'immagine di input e cerca oggetti possibili con un algoritmo di ricerca selettiva, generando circa 2000 *region proposal*, regioni da "proporre" agli strati successivi;
2. esegue una rete convoluzionale su ognuna di esse;
3. prende l'output di ogni CNN e lo dà in pasto a:
 - una **SVM** per classificare le regioni;
 - una **regressione lineare** per creare un riquadro attorno all'oggetto, se esiste.

Gli step sono mostrati nella figura 3.8.

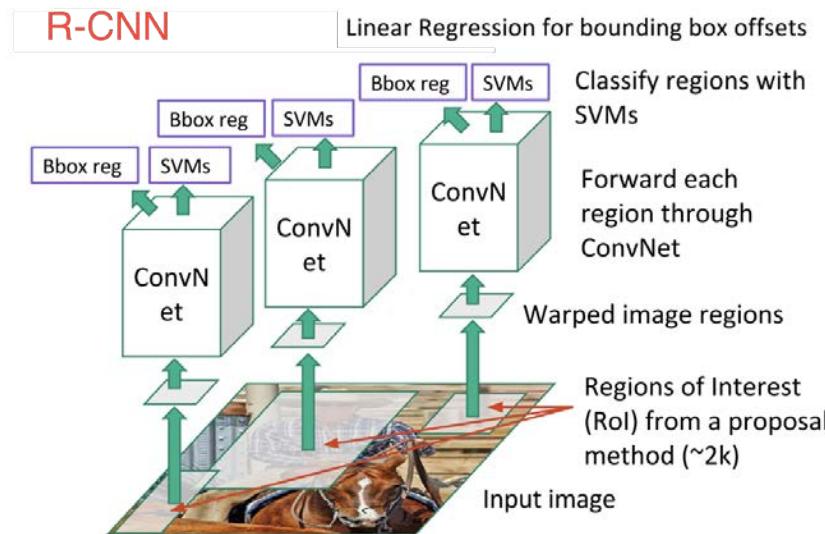


Figura 3.8: Architettura R-CNN

Intuitivamente, prima di tutto si trovano le possibili regioni in cui può essere contenuto un oggetto, poi si estraggono le informazioni ad esse associate, poi in base a queste si classificano. Si è trasformato un problema di *object detection* in uno di classificazione: era un approccio molto intuitivo ma allo stesso tempo lento e oneroso.

Fast R-CNN

Questa architettura è stata la diretta discendente della R-CNN, ma ha aumentato la sua velocità con due fattori chiave:

- esegue l'estrazione di informazioni una volta prima di creare le regioni da proporre; così facendo può permettersi di eseguire una sola CNN al posto di doverne eseguire una per ogni regione;
- sostituisce la SVM con un *layer softmax*.

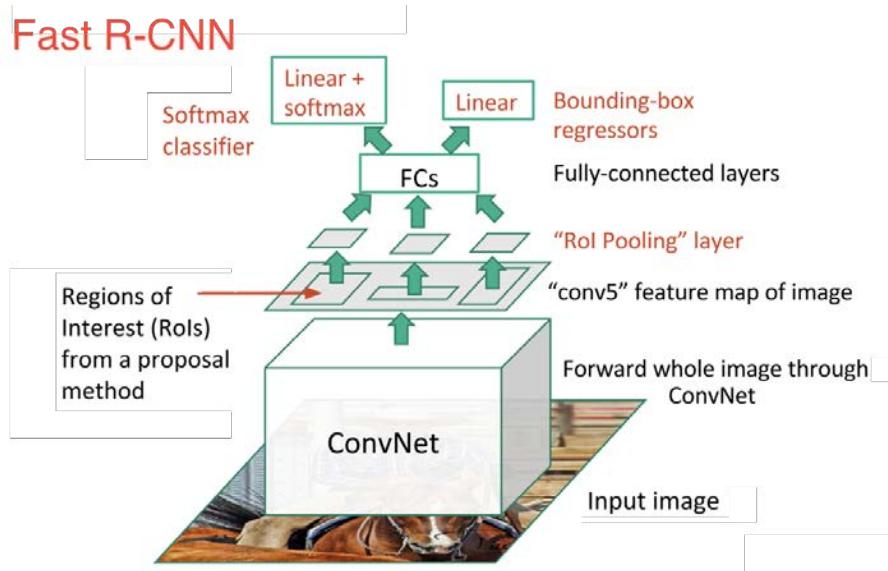


Figura 3.9: Architettura Fast R-CNN

La nuova architettura si mostrava quindi come in figura 3.9. Come si può vedere in figura, le *region proposals* sono generate a partire dall'ultimo strato di informazioni ricavate dalla rete e non dall'immagine in sé. È quindi possibile allenare una singola CNN per l'immagine intera. Inoltre, al posto di usare molte SVM per classificare ogni classe, c'è un solo *layer softmax* che genera una probabilità diretta.

La Fast R-CNN era molto più veloce della precedente, ma l'algoritmo di *selective search* era ancora troppo lento.

Faster R-CNN

Per risolvere questo ultimo collo di bottiglia si è adottato, al posto dell'algoritmo di ricerca selettiva, una rete neurale veloce. In particolare, una RPN, *Region Proposal Network*:

- viene generata una *sliding window* di dimensione 3x3 posizionata sull'ultimo strato della CNN iniziale, che si muove lungo la mappa e ne restituisce una versione più piccola, ridotta (ad esempio a 256-d);
- per ogni *sliding window* genera più possibili regioni basate su alcune *boxes* di proporzione prestabilita;
- ogni regione proposta consiste di:
 - un punteggio, *score*, col quale viene valutata la *box* trovata;
 - le effettive quattro coordinate della scatola;

In altre parole, si guarda in ogni posto proposto dall'ultimo strato della *conv net*, si considerano k diverse scatole centrate su ognuna di esse: una alta, una larga, una grande, una piccola, ecc. Per ognuna di queste viene generato un punteggio, una sorta di *likelihood*. Da notare, infine, che anche se la RPN ha come output delle *bounding*

boxes, essa non aspira a classificare un potenziale oggetto: semplicemente propone delle regioni in cui ci potrebbe essere. Se una scatola ha un punteggio sufficiente, viene quindi passata avanti come *region proposal*.

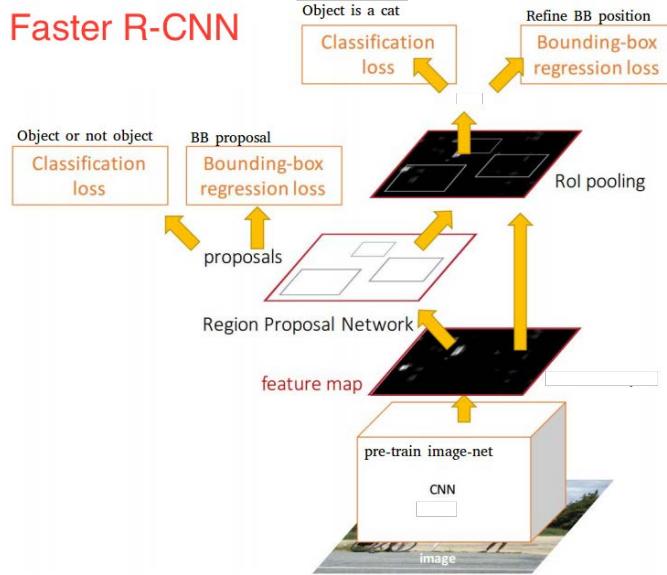


Figura 3.10: Architettura Faster R-CNN

3.4.3 SSD

Mentre la Faster R-CNN prima generava delle regioni, per poi usare un *fully connected layer* per classificarle, la SSD fa tutto questo in un solo colpo. Da qui il nome, Single-Shot Detector. In pratica, data un'immagine appartenente ad un *set* con delle *ground truth label*, ovvero delle immagini etichettate, la SSD fa questi passaggi:

1. prima passa l'immagine attraverso una serie di strati convoluzionali, raggruppando più insiemi di mappe con informazioni a diverse misure (ad esempio 10x10, poi 6x6, e così via);
2. per ogni *location*, per ognuna delle *feature maps*, usa un filtro con una convoluzione 3x3 per valutare insiemi più piccoli di scatole; questo è più o meno quello che viene fatto con le scatole della Faster R-CNN;
3. per ogni scatola, valuta contemporaneamente:
 - un offset per una possibile *bounding box*;
 - la probabilità collegata ad una classe;
4. durante l'allenamento, fa il match delle scatole trovate con quelle effettivamente etichettate, basato su **IoU**.

L'allenamento di una SSD può tuttavia essere un'impresa. Se nella Faster avevamo infatti una minima scelta basata sulla probabilità, qui le regioni proposte sono *tutte*

quelle possibili all'interno dell'immagine. Di conseguenza quasi sicuramente molte delle scatole trovate sono da scartare. Per migliorare questo aspetto sono stati introdotti due approcci:

- usare una *non-maximum suppression* per raggruppare assieme alcune scatole che sono molto sovrapposte;
- usare l'*hard negative mining* per bilanciare le classi durante l'allenamento.

3.4.4 Considerazioni finali

La SSD è una rete molto veloce e ha una performance paragonabile alle altre. Tuttavia, nonostante sia una tra le più complesse architetture e una tra le più lente, allo stato attuale la Faster è tra le migliori in termini di accuratezza¹. Per questo motivo, dopo uno studio preliminare dell'argomento, visto il *dataset* a disposizione e vista la mancanza di tempo per poter personalmente testare il modello migliore, ho deciso di utilizzare questa architettura. In seguito discuterò di come ho personalizzato questa rete, in base ai risultati sul mio particolare problema.

¹ COCO-trained models statistics. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md#coco-trained-models (visitato il 17/09/2018).

Capitolo 4

Progettazione

Una volta acquisite delle conoscenze teoriche che mi permettessero di affrontare l'argomento mi sono reso conto che avrei avuto bisogno di un algoritmo che si prendesse carico di svariati passi:

1. generare un *dataset* compatibile col linguaggio delle API di Tensorflow[32];
2. usare i pesi di una rete già allenata e che fosse compatibile con i miei scopi sfruttando il *transfer learning* per creare un grafo personalizzato con cui fare inferenza poi;
3. elaborare un file PDF su cui fare inferenza, sfruttando il grafo di cui sopra, per generare del testo puro e dei dati tabellari su CSV.

A questo scopo mi sono presto convinto di dover sviluppare due progetti ben distinti: TableTrainNet[39] e IntelligentOCR[40]. Il primo si sarebbe occupato di elaborare il *dataset* col quale avrei allenato la mia rete neurale, il secondo invece avrebbe gestito una *pipeline* che si sarebbe occupata della conversione da PDF a testo e csv.

4.1 Tecnologie e strumenti utilizzati

4.1.1 Codice e versionamento

Per entrambi i progetti è stato utilizzato il linguaggio di programmazione Python[41] versione 3. Inoltre, come raccomandato in azienda, è stato fatto l'utilizzo di PyCharm[5], un IDE per Python dell'azienda Jetbrains[©] che, oltre ad offrire un assistente "intelligente" per la scrittura di codice e *safe refactoring*, include anche degli strumenti di sviluppo quali *debugging* e *deploying* su GitHub[6]. Si è fatto uso di quest'ultimo strumento per il versionamento del codice, che può essere trovato alle *repositories* citate in bibliografia.

Da notare, infine, che tutto il codice è regolabile da comodi files contenenti solamente "costanti". Queste "costanti", che poi non sono altro che le variabili che l'utente può e deve personalizzare per far funzionare il prodotto sulla propria macchina, si occupano di gestire tutto il funzionamento del prodotto, gestendo gli input, i modificatori e gli output.

4.1.2 Allenamento di reti neurali e inferenza

Per tutta la parte di intelligenza artificiale mi sono affidato al framework Tensorflow[32]. Tensorflow è una libreria *open source* per computazione numerica ad alte prestazioni. Mi sono reso conto quasi subito che non avrei avuto il tempo di costruire una rete personalizzata per mancanza di tempo. Tra i prodotti "pre-confezionati" e che mi avrebbero permesso poi una certa personalizzazione senza dover avere troppo a che fare con del codice "a basso livello" è stata per l'appunto l'API di *object detection*[37]. Questa libreria permette, utilizzando il *core* di Tensorflow, di allenare e valutare reti neurali a partire da file di configurazione personalizzati. Alcuni parametri salienti che sono modificabili sono, per esempio:

- il *learning rate* e le varianti ad esso associate;
- la *batch size*;
- la tipologia di *optimizer*;
- alcune opzioni di *data augmentation*;
- i *checkpoint* di partenza, da quali prendere i pesi di altre reti neurali per approfittare del *transfer learning*.

Questi sono solo alcuni dei parametri messi a disposizione e dei quali parlerò più profusamente durante la descrizione del progetto TableTrainNet. Da notare infine che assieme a questa libreria sono stati messi a disposizione anche dei pesi di partenza, ovvero le reti già pre-allenate che poi ho utilizzato nel corso del mio progetto. Inoltre Tensorflow mette nativamente a disposizione uno strumento di controllo automatico dell'andamento dell'apprendimento della rete chiamato Tensorboard.

Per quanto riguarda l'inferenza, Tensorflow non si è dimostrato il framework dal linguaggio più semplice da utilizzare - Keras[31] utilizza una sintassi molto meno complessa -. Tuttavia, la documentazione online è molto buona e l'utilizzo del framework per ottenere delle *boxes* a partire da immagini è standard e ben comprovato.

4.1.3 *Editing* di immagini

Il progetto si è principalmente basato sulle immagini: avevo bisogno di strumenti efficaci che mi permettessero di perdere il meno tempo possibile nelle impostazioni, per poterlo poi impiegare nella valutazione dei risultati.

Pillow

Ho trovato in Pillow[42] veramente rapido nell'utilizzo per ottenere immagini dal disco o da *arrays*, farne dei ritagli, convertirle in scala di grigi e poi salvarle sul disco.

OpenCV

Per quanto riguarda invece modificazioni un po' più complesse ho utilizzato invece OpenCV[43], che lavora su immagini sotto forma di array ed è più orientata al *machine learning* e al *computer vision*.

Alyn con Scikit-image

Ho inoltre utilizzato uno strumento di raddrizzamento automatico delle immagini, Alyn[44], modificando un progetto che avevo trovato valido, correggendolo e adattandolo alle mie esigenze. Questo progetto fa uso di Scikit-image[45].

4.1.4 Estrazione di informazioni

Da PDF ad immagini

pdftoppm[46] si è dimostrato il miglior strumento per estrarre immagini a partire da PDF. Inizialmente avevo usato un *wrapper* attorno a ImageMagik[47], ma con PDF di grandi dimensioni il programma non riusciva a gestire i files temporanei, causando un intasamento del computer in cui era eseguito.

Da PDF a tabelle

Tabula[48] è un'applicativo Java che si occupa di interpretare un foglio PDF per estrarre tutte le possibili tabelle ed esportarle in un CSV, utilizzando pandas[49].

Da immagini a testo

Per estrarre il testo a partire da immagini ho usato Tesseract[50]. Questo è un motore OCR per il riconoscimento di caratteri all'interno di un'immagine, che è stato sviluppato inizialmente ai laboratori in Bristol di Hewlett-Packard. È stato reso pubblico nel 2005 e poi sviluppato da Google. Ho usato la versione 4, attualmente in sviluppo attivo, che fa uso di una rete neurale LSTM per migliorare l'individuazione dei caratteri. Non mi sono occupato di allenare questa rete, ma ho usato l'allenamento offerto direttamente da Google.

4.2 TableTrainNet: creare il *dataset*

Questo progetto è stato sviluppato per creare una rete neurale che riconosca tabelle all'interno di documenti. Il prodotto di questo progetto serve poi ad IntelligentOCR per funzionare.

4.2.1 *Overview* generale

Il progetto usa una delle reti neurali offerte^[51] da Tensorflow per analizzare un *dataset* contenente immagini con tabelle. In più è stato personalizzato un file di configurazione per l'allenamento e valutazione per utilizzare l'API di *object detection*^[37].

4.2.2 Il *dataset* iniziale

La prima difficoltà di creare un buon *dataset*, inutile a dirsi, è trovare una buona base. Sebbene siano molto comuni insiemi di immagini relative ad oggetti generici, sono altresì rari quelli relativi agli oggetti che si possono trovare nei documenti. È stato quindi inizialmente complesso trovare delle immagini con documenti già etichettati, senza dovermeli creare da solo. Fortunatamente sono entrato in contatto con una competizione online di POD[52] organizzata dall'Università di Pechino, Istituto di Computer Science and Technology[53]. Il *dataset* è formato da 2000 documenti in inglese selezionati da 1500 articoli scientifici di CiteSeerX[54], contenenti sia tabelle che formule e grafici. Mostra poi una buona varietà sia per quanto riguarda il *layout* della pagina, sia per lo stile degli oggetti, includendo testo a singola e doppia colonna.



Figura 4.1: Esempio di immagini presenti nel dataset

Per quanto riguarda il mio progetto ho preso in considerazione solamente le tabelle, ignorando le altre informazioni che si possono trovare. Dovendo infatti orientare la mia rete neurale ad analizzare polizze assicurative, ho preferito evitare di inserire elementi di disturbo - ad esempio, rilevazioni errate di oggetti non presenti - al fine di migliorare le *detections*.

Nel mio caso ho deciso di dedicare il 60% del *dataset* al *training* e il 40% al test. Purtroppo lo strumento offerto da Tensorflow non permetteva l'aggiunta della terza e molto importante parte della validazione, che serve per valutare l'affidabilità dei risultati ottenuti con la fase di test. Inoltre si può notare che ho dedicato molto spazio al test: il mio *dataset* non è così grande, allora ho pensato di dedicare più immagini alla parte di test così da ridurre l'*overfitting* nel quale altrimenti sarei potuto incappare.

4.2.3 Modifica delle immagini per migliorare l'apprendimento

Per allenare in maniera efficace la rete neurale bisogna ricordare il pre-allenamento eseguito dalle reti offerte da Tensorflow. Queste infatti sono allenate sui *dataset*:

- COCO dataset[55];
- Kitti dataset[56];
- Open Images dataset[57];
- AVA v2.1 dataset[58];

Tutti questi *dataset* trattano immagini a tre canali di colore e che contengono oggetti "uniformi", per così dire. È possibile immaginare quindi che la rete abbia imparato che c'è una sorta di "continuità" fra gli oggetti che poi sono identificati come tali. Al contrario, ho supposto che noi umani leggiamo le tabelle principalmente grazie ad una informazione spaziale: ovvero, sappiamo che la tabella è fatta così perché è composta da varie celle, possibilmente allineate.

Non essendo sicuro che questa informazione fosse facilmente recepibile dalla rete offerta da Google, ho cercato alcune soluzioni e mi sono imbattuto in un articolo scientifico, *Table Detection using Deep Learning*[59]. In questo articolo in particolare suggerivano di eseguire la seguente trasformazione[60] sulle immagini con le tabelle:

- trasformazione euclidea della distanza sul canale blu;
- trasformazione lineare della distanza sul canale verde;
- trasformazione sul massimo della distanza sul canale rosso;

Per trasformazione della distanza si intende la rappresentazione della distanza tra il pixel a zero più vicino per ogni pixel dell'immagine. Il risultato di questa trasformazione si può apprezzare all'immagine 4.2. Da notare che le immagini rispettano le proporzioni e le posizioni.

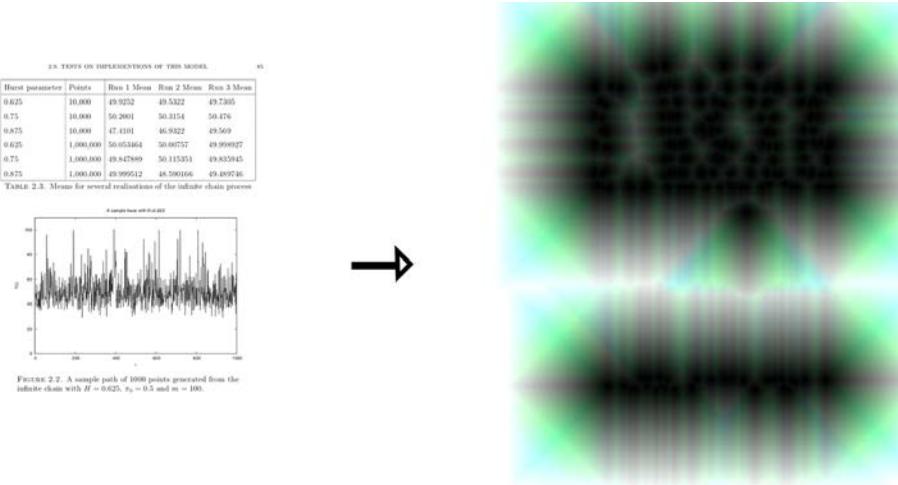


Figura 4.2: Esempio di trasformazione dell'immagine con il calcolo delle distanze

Innanzitutto con questa trasformazione si possono annidare più informazioni - relative alle tre trasformazioni - su tre canali. Poi possiamo vedere come l'immagine assomigli di più ad una che la rete neurale pre-allenata si aspetterebbe: ovvero con una distribuzione semi omogenea di pixel, all'interno dei quali cercare le ricorrenze del caso. Possiamo infatti vedere come il grafico sia scarsamente rappresentato, poiché la quantità di pixel "colorati" è mediamente bassa. Lo è di più nella sua descrizione, ad esempio; inoltre intravediamo una certa organizzazione nella trasformazione della tabella: proprio quello che stavamo cercando!

4.2.4 Preparazione delle etichette

Le etichette del *dataset* sul quale ho lavorato erano sotto forma di XML, uno per ogni immagine, all'interno del quale erano presenti le informazioni riguardanti le *boxes* che racchiudono gli oggetti presenti nelle immagini. Poiché Tensorflow accetta solamente formati pandas[49] per le etichette, ho provveduto a trasformare le informazioni in due CSV distinti, uno per il training e l'altro per il test.

4.2.5 Creazione dei file *TFRecord*

Creare i file *TFRecord*, ovvero dei files di input che Tensorflow possa capire, avendo immagini ed etichette pronte, è abbastanza facile. È bastato infatti personalizzare lo script[61] offerto direttamente dall'API di Google.

4.2.6 Allenamento della rete neurale personalizzata

Come già anticipato è stato possibile non concentrarsi sul codice da scrivere per personalizzare la rete e riuscire a gestirla da un file di configurazione. I parametri - o, altresì chiamati nel gergo, *hyperparameters* -, sono tanti e ognuno ha un'importanza maggiore o minore rispetto agli altri. Inoltre ho avuto la possibilità di scegliere da quale tipologia di architettura di rete neurale partire.

L'allenamento è stato effettuato con la mia macchina, della quale caratteristiche scriverò brevemente:

- *CPU*: Intel Core i5 8300H, 3.80GHz in Turbo Mode;
- *GPU*: NVidia GTX1050 GDDR5 4GB RAM;
- *RAM*: 8GB DDR4 2667MHz;
- *Storage*: SSD NVMe 256GB

Scelta della rete di partenza

Per quanto riguarda la scelta dei pesi iniziali, ovvero della rete neurale di partenza, non avendo molto tempo a disposizione ho deciso di effettuare una decisione a priori guardando le statistiche[38] degli allenamenti delle reti basati sul *dataset* COCO[55]. La rete con risultati migliori a parità di velocità si è rivelata essere la faster_rcnn_inception_v2_coco, ho quindi basato tutti i miei successivi esperimenti su questa.

Scelta dei parametri da personalizzare

Per quanto invece concerne la scelta dei parametri, ho deciso di focalizzare la mia attenzione su alcuni di questi:

- *image_resizer*, che si occupa della dimensione e dei protocolli di ridimensionamento delle immagini;
- *l2_regularizer*, ovvero il peso della trasformazione L2[62];
- *first_stage_nms_iou_threshold* e *iou_threshold*, ovvero il peso dato all'IoU;
- *optimizers* e *learning_rate*, ovvero la tipologia di *optimizer*[63] utilizzato e come gestire il *learning rate*.

Ho scelto di personalizzare solo questi principalmente per via della letteratura e dei corsi che ho seguito: questi infatti consigliavano di concentrarsi su questi parametri innanzitutto, poi di esplorare gli altri. Non avendo a disposizione molto tempo per testare ho quindi pensato di effettuare questa scelta.

Esempio di utilizzo di Tensorboard

Tensorboard è stato un valido alleato nell'allenare le reti neurali. Durante l'apprendimento permette di vedere l'andamento di due curve: una in arancione relativa al *training*, una in blu relativa alla valutazione - che poi è quella più rilevante come "risultato". In particolare, i grafici che più ho tenuto sotto controllo sono stati:

- la *DetectionBoxes_Precision/mAP (large)*, principalmente perché la maggior parte delle scatole da individuare corrispondevano a dimensioni generose, essendo le tabelle relativamente grandi rispetto all'immagine. Nell'immagine 4.3 possiamo vedere la valutazione di una rete con *adam optimizer*;
- La *loss*, per valutare se il *training* o la valutazione stessero andando in *overfitting*. Nell'immagine 4.4 possiamo vedere l'andamento della perdita per una rete neurale con *momentum optimizer*.



Figura 4.3: Esempio di schermata classica con Tensorboard

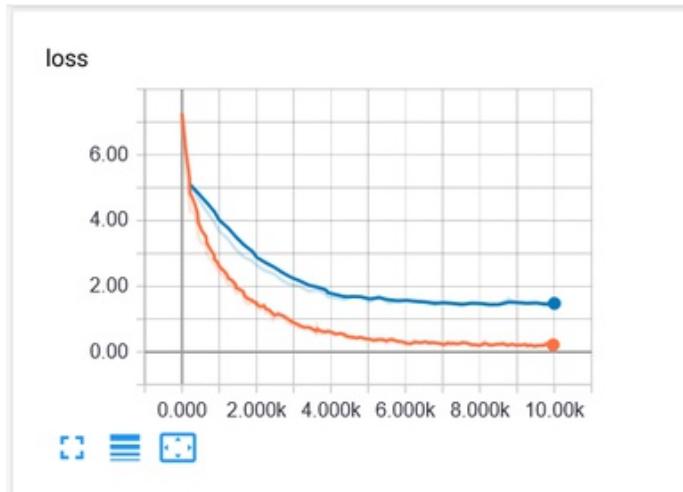


Figura 4.4: Esempio di loss con *Momentum Optimizer*

Considerazioni sui parametri

image_resizer

La personalizzazione di questo parametro è stata piuttosto semplice: ho dovuto trovare un buon compromesso tra grandezza dell'immagine da passare come input e la quantità di memoria a disposizione. Alla fine la scelta di ridimensionare in maniera fissa (quindi non rispettando le proporzioni dell'immagine) ogni immagine come input a 400x400px mi ha permesso una buona risoluzione a fronte di un'occupazione decente in RAM. Da notare che non rispettare le proporzioni dell'immagine - ovvero forzare una risoluzione predefinita - mi ha permesso di aumentare la *batch_size* oltre il valore predefinito di 1-

l2_regularizer

L'impostazione di questo parametro è correlato alla presenza di *overfitting* nel proprio modello. Grazie a tensorboard ho potuto tenere d'occhio l'andamento degli apprendi-

menti e, dopo molti tentativi, ho capito che probabilmente il valore che più si addice al mio *dataset* è di $1 * 10^{-4}$ come valore iniziale, che poi aumenta fino a $4 * 10^{-4}$.

first_stage_nms_iou_threshold* e *iou_threshold

Questi due parametri gestiscono la metrica dell'IoU. Ho deciso di mettere mano anche a questo parametro dopo aver visionato il comportamento delle reti neurali quando testate (sezione A.1). Infatti mi sono reso conto che molto spesso la rete neurale confondeva una parte delle tabelle che avrebbe dovuto trovare per la tabella intera. Questo fatto è dovuto probabilmente alla diversità delle immagini del *dataset* utilizzato per il training e di quelle che poi effettivamente sono presenti all'interno delle polizze. Un modo per mitigare questo effetto è stato quindi quello di abbassare il valore di questi due parametri a 0.5.

optimizers* e *learning_rate

La scelta di questi due parametri è stata dettata dalla disponibilità dell'API di Google, dalla presenza di parametri di default e dalla letteratura. In particolare la disponibilità di *optimizers* da parte dell'API è di:

- *RMSPropOptimizer*[64];
- *MomentumOptimizer*[65];
- *AdamOptimizer*[66].

La scelta predefinita era impostata sul *Momentum*. Tuttavia nei corsi seguiti su Coursera era emerso che l'*Adam* avesse una migliore performance, a patto di regolare correttamente il *learning rate*. Ho eseguito alcune prove con il *momentum*, poi ho deciso di cambiare. Quindi, dopo aver impostato l'*Adam* e il relativo *learning rate* con la seguente formula (dell'*exponentially decay learning rate*), dati l_{r0} *learning rate* iniziale, t il numero di iterazioni e k un parametro:

$$l_r = l_{r0} * e^{(-k*t)}$$

e dopo molte prove - per evitare un apprendimento troppo veloce oppure troppo lento -, sono arrivato ad impostare:

$$l_{r0} = 1 \exp^{-4}$$

$$k = 0.8$$

$$\text{decay_steps} = 300$$

dove *decay_steps* sono i passi dopo i quali valutare il nuovo *learning rate*.

Interpretazione e modifica delle *boxes*

La rete neurale è per lo più rimasta in allenamento per 7000 iterazioni per un tempo variabile tra le 4 e le 8 ore di tempo. Una volta esportato il grafo congelato, così da essere pronto per l'inferenza, ho potuto testare le varie configurazioni. In A.1 troviamo i risultati visivi di quelle provate.

Ci si rende subito conto di come l'identificazione delle tabelle non sia precisa. Soprattutto si nota come il più delle volte l'algoritmo confonde una parte per l'insieme, identificando solamente porzioni di tabella e spesso in maniera sovrapposta tra una identificazione e l'altra. Inoltre anche i punteggi non sono mai eccelsi: non appena si

superà la valutazione di 0.8 l'identificazione inizia ad essere lacunosa.

Ho quindi pensato ad una soluzione. Dal momento che conoscevo la tipologia di documenti che avrei dovuto analizzare, ho stilato le seguenti osservazioni:

- ogni tabella compare sempre senza testo attorno e a singola colonna;
- non ci sono mai più di quattro tabelle nella stessa pagina;
- le tabelle sono in formato misto, ma per lo più composto di parole e numeri.

Da queste informazioni ho dedotto di poter:

- tagliare la pagina non attorno alla scatola trovata ma su tutta la larghezza di quella. Così avrei diminuito l'errore dovuto all'individuazione, dovendo considerare solo due punti invece di quattro forniti (posizione di minimo e di massimo sull'asse verticale);
- unire le informazioni di alcune scatole con un punteggio abbastanza alto (personalmente impostato a 0.6) che venivano tra loro sovrapposte, generando quindi un'unica grande *box*.

Nella sezione A.1 si potranno apprezzare le nuove scatole che, in rosso, individuano quindi la zona che si dovrà andare a tagliare.

4.3 IntelligentOCR

4.3.1 Overview generale

IntelligentOCR è l'effettivo programma che la mia azienda può utilizzare in maniera produttiva. Si occupa di distinguere le tabelle all'interno di documenti PDF - nel mio particolare caso, in polizze assicurative -, per poi poter processare in maniera diversa le une e il testo rimanente. Quindi, questo programma:

1. prende come input un file PDF e lo trasforma in immagini;
2. per ogni immagine, ricerca tabelle al suo interno;
3. processa ogni tabella con un estrattore di tabelle da documenti, così da restituirne la struttura e da restituirla su CSV;
4. processa le immagini con testo puro al loro interno per estrarre del testo su TXT.

Il focus durante la progettazione di questo programma è stato la leggibilità del codice e la leggerezza in RAM, poiché il programma è destinato a funzionare su un server con relativamente poca memoria a disposizione e ad essere manutenuto e migliorato da altre persone.

4.3.2 Design pattern utilizzati

Tutti i passaggi di cui sopra mi hanno indotto a pensare ad una sorta di "catena di montaggio" per il mio prodotto, che lo rendesse facilmente migliorabile e testabile. Questo soprattutto in un'ottica aziendale: difficilmente mi occuperò di manutenere il prodotto che ho creato e un'altra persona dovrà riuscire a manuterlo senza fatica.

Essendo una sequenza di azioni non correlate fra loro ho pensato che il *Pipeline Pattern*[67] potesse venirmi in aiuto. Infatti questo garantisce un'ottimo approccio per testare e per applicare il *single responsibility principle*[68], in quanto ogni pezzo di codice fa solamente la sola o le due uniche cose per le quali è stato progettato, dopodiché passa il testimone al pezzo successivo. Non da meno, aumenta in maniera esponenziale la leggibilità del programma.

Non da ultimo è da notare che è stato eseguito un approccio *bottom-up*, ovvero dal basso verso l'alto. Ho cercato di costruire prima tutti i passaggi in maniera autonoma, senza che gli uni dipendessero dagli altri; ho poi provveduto alla composizione dei vari pezzi. Si può notare questo approccio nella composizione del file *pipeline.py*, dove non vengono effettuate operazioni aggiuntive ma vengono solamente invocati i vari anelli della catena.

4.3.3 La pipeline

Poiché il prodotto si configura proprio come una *pipeline*, ne mostrerò la progettazione per parti. Si può apprezzare il fatto che ogni step è separato dagli altri in maniera stagna, tant'è che si può attivare l'output su disco oppure testare con documenti o immagini di prova in qualunque punto della catena di montaggio.

Da PDF ad immagini

Un PDF è di per sé un formato prestante: può anche contenere delle informazioni relative al layout che poi possono essere utilizzate da lettori più o meno avanzati. Per questo motivo se un PDF non è frutto di una scansione, ma anzi è stato "stampato" digitalmente, allora tutto ciò che riguarda l'inferenza con rete neurali, la suddivisione in tabelle e testo, non ha più senso d'esistere perché esistono già molti strumenti che si occupano di interpretare tabelle. Tuttavia l'azienda mi ha esplicitamente chiesto di coprire il caso generale, ovvero di PDF provenienti da scansioni. In effetti si è poi capito che la stragrande maggioranza delle polizze assicurative presenti nel database era composto da PDF di siffatta natura.

Estrazione Ho quindi inizialmente provveduto all'utilizzo di Wand^[69], *wrapper* attorno a ImageMagik^[47], che esportava tutte le immagini in memoria, per poi poterle utilizzare nel resto della catena di montaggio. Tuttavia, dopo aver testato in maniera più approfondita il prodotto, mi sono reso conto che la memoria veniva occupata per centinaia di MB inutilmente, in quanto venivano caricate tutte le immagini del documento per poi comunque processarne una alla volta. Inoltre ImageMagik è affetto da un bug che, in caso di PDF di grandi dimensioni, fa sì che non liberi lo spazio temporaneo occupato su hard disk, portando all'occupazione di decine di GB inutilmente. Tutto ciò mi ha portato a cambiare approccio: innanzitutto ho proceduto all'utilizzo di pdftoppm^[46] da riga di comando direttamente da Python. Questo mi ha permesso di accedere una pagina alla volta ad ogni PDF, quindi di avere un'occupazione in RAM minima. Inoltre mi sono informato circa l'utilizzo dei *generators* di Python, regolando quindi l'accesso ad ogni pagina solo quando effettivamente necessario.

Da questo punto in poi tutti i passaggi vengono eseguiti su una pagina alla volta.

Miglioramento dell'input Una volta ottenuta la pagina possono essere eseguiti dei miglioramenti sull'input. In questo progetto viene solamente applicato un raddrizzamento attraverso Alyn^[44], ma è in questa sede che è possibile applicare altre trasformazioni. Una possibilità studiata ma non applicata in quanto non richiesto e per mancanza di tempo sarebbe stata la riduzione del rumore di fondo ispirandosi a dei *kernels*^[70] offerti da Kaggle^[71].

Da immagini a ritagli

In questo stadio viene eseguita l'inferenza con la rete neurale descritta nella sezione 4.2.6. Qui vengono replicati gli stessi passi: viene convertita l'immagine in un formato più "gradevole" per la rete neurale, vengono emesse le *boxes* e vengono interpretate. A questo punto viene eseguito il taglio vero e proprio: si crea quindi una lista di immagini, in ognuna delle quali vengono memorizzate tutte le tabelle ritagliate dalla pagina originale, e un'immagine unica in cui vengono uniti tutti i ritagli rimanenti.

A questo punto la lista di immagini e quella singola con tutto il testo vengono processate in maniera diversa.

Da tabelle a CSV

Ogni tabella della lista viene convertita ciascuna in un file PDF ricercabile con Tesseract^[50]. Questo è infatti l'unico formato che Tabula^[48] accetta come input, per poter poi esportare la struttura calcolata.

Da immagini a testo puro

Ogni collage di immagini rimanenti dopo l'estrazione delle tabelle viene invece processato semplicemente da Tesseract e viene prodotto un file TXT con all'interno tutto il testo che lo strumento è riuscito a comprendere.

4.3.4 Risultati ottenuti

I risultati si possono visionare alla sezione [A.2](#). Si può notare come non siano ancora ottimali. Come si può osservare alla sezione [A.2.4](#) nel caso di documenti ben formati il riconoscimento è sufficiente; tuttavia, se i documenti sono scansionati male, l'algoritmo funziona molto peggio (ad esempio in sezione [A.2.3](#)). È tuttavia un passo in avanti notevole nella distinzione tra i due tipi di contenuti, sui quali poi si potranno eseguire altri tipi di analisi.

Capitolo 5

Valutazione retrospettiva

Si riporta di seguito una valutazione retrospettiva dell'esperienza di stage, al fine di valutare ciò che è stato fatto, le esperienze apprese e trarre insegnamenti per il futuro.

5.1 Raggiungimento degli obiettivi

Durante tutto il tempo di stage mi sono reso conto di dover continuamente tenere in considerazione le conoscenze da acquisire e la loro effettiva implementazione. La teoria da imparare è molta e non ho esaurito lo scibile in così poco tempo: ogni volta che mi si poneva un problema ho dovuto scegliere se risolverlo con un approccio teorico - e dispendioso in termini di tempo - oppure con uno funzionale, ma che non sempre permette di trarre insegnamenti validi. D'altro canto, il rischio opposto - ovvero di fossilizzarsi sulla teoria - era altrettanto pericoloso. Nonostante ciò ho raggiunto una buona copertura degli obiettivi definiti nel piano di lavoro.

ID	Descrizione	Stato
Obbligatori		
O01	Creazione di un algoritmo che faciliti la creazione di un dataset per allenare una rete neurale	Completato
O02	Creazione di una struttura di testing per la valutazione visiva dei risultati ottenuti	Completato
O03	Creazione di un algoritmo di estrazione di informazioni mirate da un libretto assicurativo	Completato
O04	Gestione di tutti gli algoritmi il più automatica possibile, possibilmente tramite l'uso di costanti	Completato
O05	Utilizzo del linguaggio Python versione 3	Completato
Desiderabili		
D01	Creazione di un batch multithreading per l'estrazione parallela di informazioni da più polizze	Completato
Facoltativi		
F01	Integrazione del sistema nell'infrastruttura dell'azienda	Non completato

Tabella 5.1: Tabella raggiungimento degli obiettivi

5.2 Resoconto dell'analisi dei rischi

Si riporta inoltre il riscontro dell'analisi dei rischi analizzati ad inizio stage.

Descrizione	Verificato	Piano attuato
Non conoscenza del linguaggio: prima di iniziare lo stage non avevo mai programmato in Python;	SI	Ho trovato dei meccanismi di apprendimento rapido del linguaggio, tramite corsi online e provando fin da subito a scrivere del codice per conto proprio. Inoltre ho chiesto spesso agli altri stagisti qualche domanda più tecnica che mi potesse risparmiare del tempo per la ricerca
Non conoscenza della materia: Prima dell'inizio dello stage la conoscenza teorica in mio possesso di <i>machine learning</i> , <i>deep learning</i> e di reti neurali era generale e per niente applicata	SI	Ho frequentato fin dalla prima settimana dei corsi specializzanti.
Non conoscenza degli strumenti: Per il <i>machine learning</i> e il <i>deep learning</i> esistono tutta una serie di strumenti già pronti all'uso che risparmiano molto tempo in fase di progettazione e codifica.	SI	Anche durante lo studio di preparazione ho avuto modo di confrontarmi con le tecnologie che poi sarei dovuto utilizzare
Esami durante il lavoro: Lo stage è iniziato senza avere la certezza di aver superato un esame, che avrei dovuto quindi sostenere durante il periodo di lavoro.	SI	Ho pianificato la progettazione in maniera tale per cui ho lasciato la fase di allenamento e test nel periodo subito antecedente l'esame. Così la sera sono riuscito a studiare proficuamente.

Tabella 5.2: Riscontro dell'analisi dei rischi

5.3 Possibili miglioramenti futuri

La scarsa conoscenza iniziale della teoria e degli strumenti non mi hanno permesso di ottimizzare al meglio il prodotto da me costruito. Sicuramente l'allenamento della rete neurale, fatto sulla macchina personale, è migliorabile. L'azienda dispone di uno spazio [AWS](#) che potrebbe essere sfruttato a questo scopo.

C'è inoltre da segnalare la diversità dei *dataset* di allenamento e di effettivo utilizzo: la creazione di un *dataset* personalizzato è d'obbligo, in questo senso, per migliorare le prestazioni della rete. A questo scopo l'idea futura è quella di implementare l'algoritmo da me creato nella piattaforma, cosicché siano gli utenti stessi a correggere e generare

il *dataset* in maniera assistita.

Altro punto da segnalare è modificare la *pipeline* di inferenza in maniera tale che essa abbia la possibilità di evitare l'algoritmo di inferenza qualora il PDF sia riconosciuto come "stampato" e non scansionato da scanner.

5.4 Conoscenze acquisite e valutazione finale

Tra le conoscenze meramente tecniche ho avuto modo di acquisire un nuovo linguaggio di programmazione e moltissimi *framework* che mi saranno estremamente utili nel percorso di studi magistrale e in prospettiva lavorativa. In particolare, l'aver imparato un po' della teoria che sta dietro i più famosi algoritmi di *deep learning* e aver messo "le mani in pasta" per quanto riguarda Tensorflow, OpenCV e Keras ha centrato appieno le aspettative che avevo in mente ad inizio stage. Ho felicemente constatato che le capacità acquisite durante il corso di studi triennale mi hanno permesso di adattarmi perfettamente e senza troppe difficoltà alla mancata conoscenza di praticamente tutto ciò che riguardava lo stage.

Non da meno ho imparato ad introdurmi in un team già preconfigurato dove il *core* aziendale non è strettamente quello informatico, bensì quello assicurativo. In più ho potuto condividere moltissime curiosità, osservazioni e possibili sviluppi ulteriori riguardanti le reti neurali con un collega lavoratore presso RiskApp, Luca Bizzaro, che mi hanno stimolato enormemente per uno studio ulteriore, più approfondito e mirato dell'argomento.

In conclusione l'esperienza presso RiskApp è stata estremamente proficua sotto entrambi gli aspetti umani e professionali e spero di poter ripetere l'esperienza anche in futuro.

Appendice A

Appendice

Qui di seguito sono presentati tutti gli allegati di cui si è discusso lungo il documento.

A.1 *Boxes* rilevate nei test

In questa sezione verranno mostrate le immagini di test con le scatole che la rete neurale ha trovato in essi. Per ogni test verranno presentate l'immagine originale e, a coppie, un'immagine con tutte le migliori scatole trovate sopra il punteggio di 0.6 e la sua corrispondente con le scatole unite secondo l'algoritmo citato a [4.2.6](#).

A.1.1 Test 0

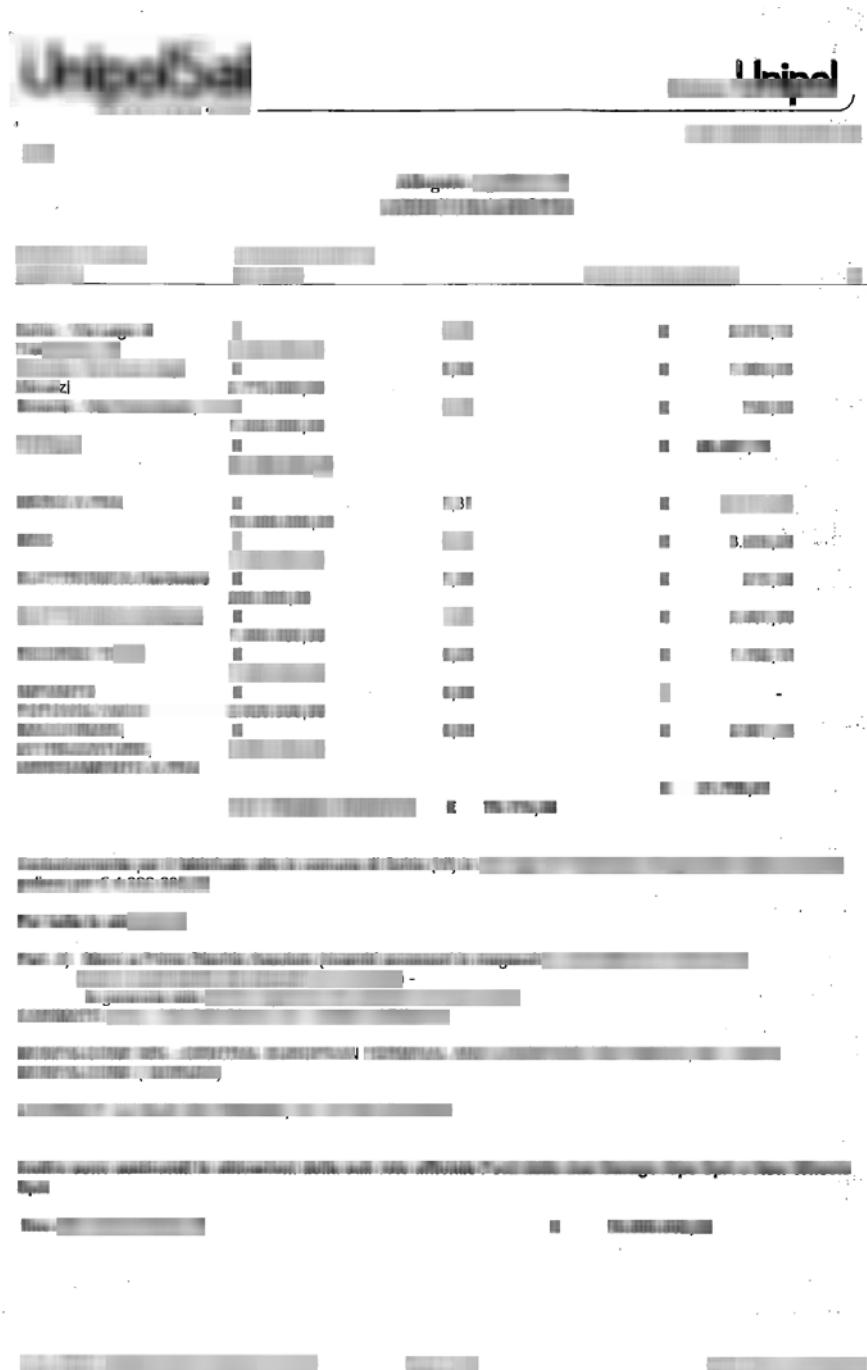


Figura A.1: Test 0

**Figura A.2:** Test 0, configurazione 1

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.008
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: .00008
                schedule {
                    step: 4500
                    learning_rate: .00004
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

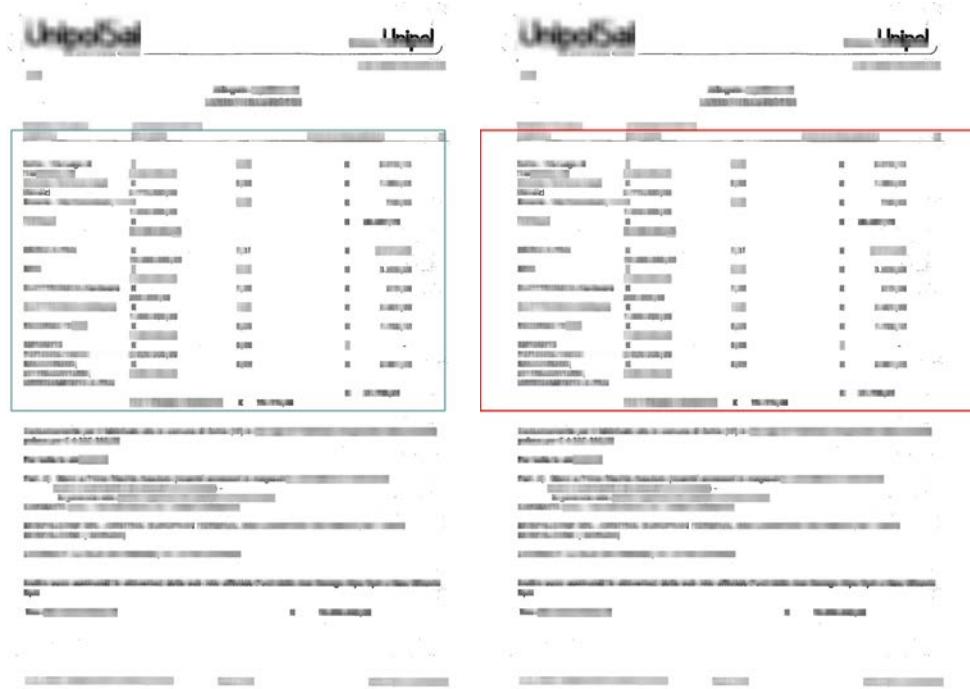


Figura A.3: Test 0, configurazione 2

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
        }
    }
}
use_moving_average: false
}

```

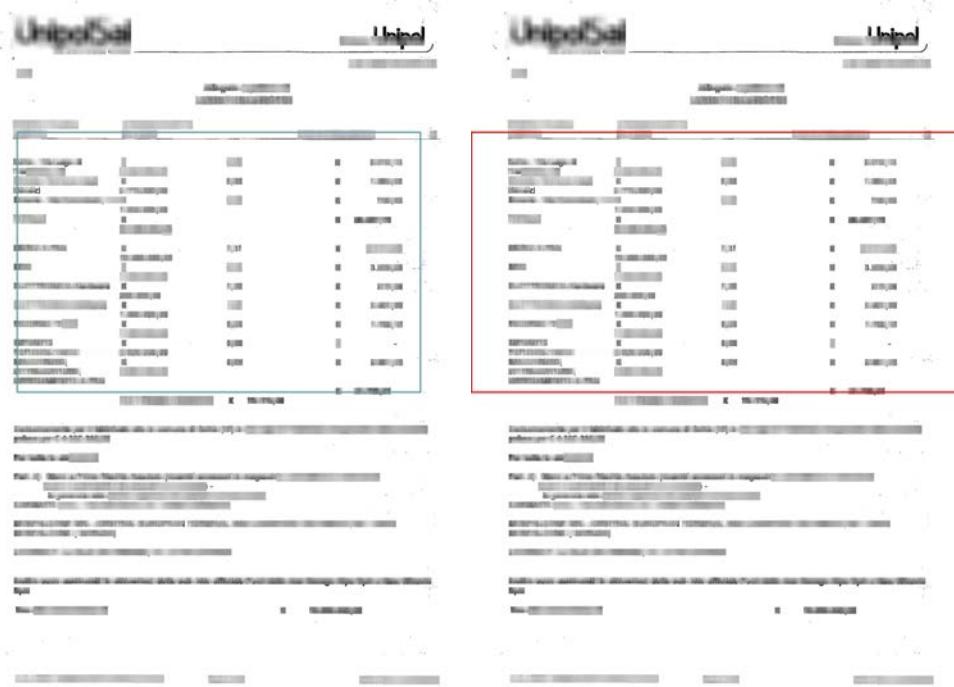


Figura A.4: Test 0, configurazione 3

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
        }
    }
}
use_moving_average: false
}
```



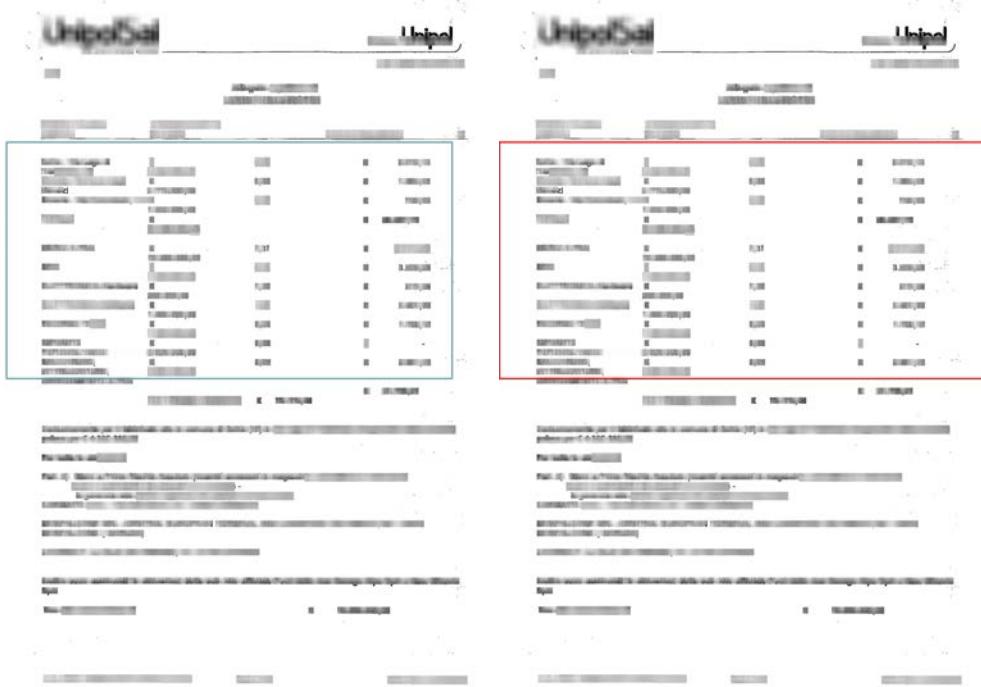
Figura A.5: Test 0, configurazione 4

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0008
                schedule {
                    step: 4500
                    learning_rate: .0008
                }
                schedule {
                    step: 7000
                    learning_rate: .0004
                }
                schedule {
                    step: 10000
                    learning_rate: .00008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

**Figura A.6:** Test 0, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

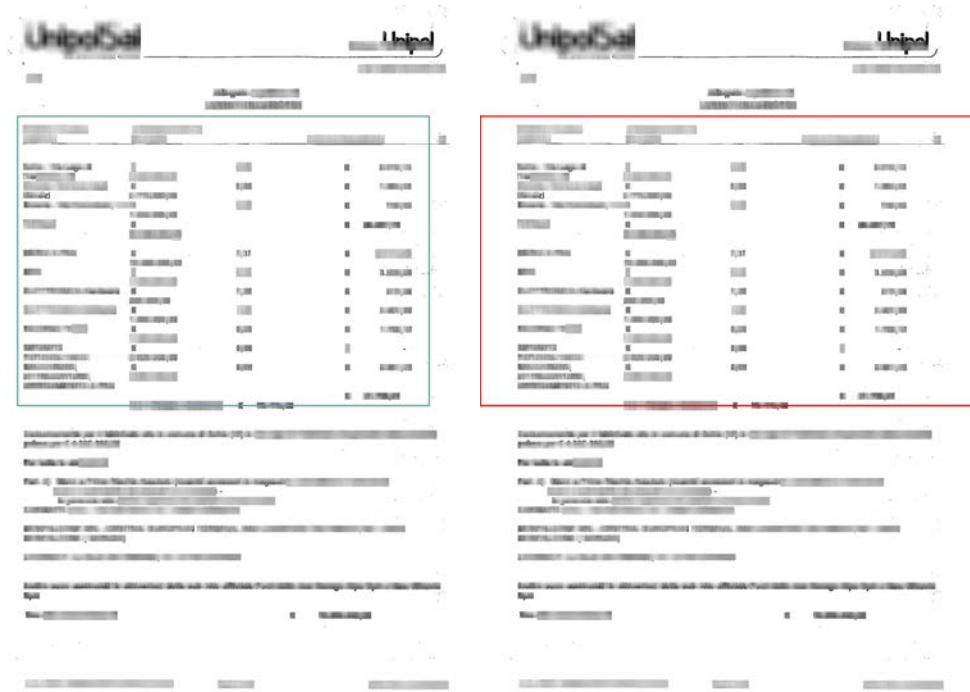


Figura A.7: Test 0, configurazione 6

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
            }
            schedule {
                step: 4500
                learning_rate: .0002
            }
            schedule {
                step: 7000
                learning_rate: .00002
            }
            schedule {
                step: 10000
                learning_rate: .000002
            }
            ...
        }
        momentum_optimizer_value: 0.9
    }
    use_moving_average: false
}

```

A.1.2 Test 1

L

[REDACTED]

[REDACTED]	[REDACTED]	<i>Tasso %</i>	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]	[REDACTED]	<i>Tasso %</i>	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

[REDACTED]

[REDACTED]

[REDACTED]	[REDACTED]	<i>Tasso %</i>	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]
[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

[REDACTED]

Figura A.8: Test 1

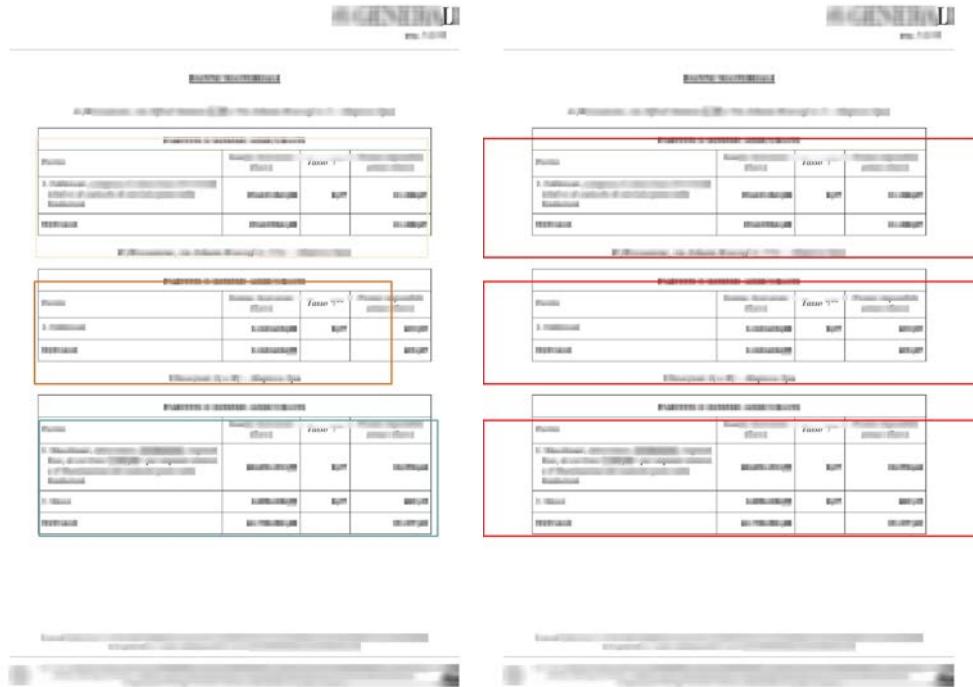


Figura A.9: Test 1, configurazione 1

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.008
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: .00008
                schedule {
                    step: 4500
                    learning_rate: .00004
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000008
                }
                ...
            }
            momentum_optimizer_value: 0.9
            use_moving_average: false
        }
    }
}

```

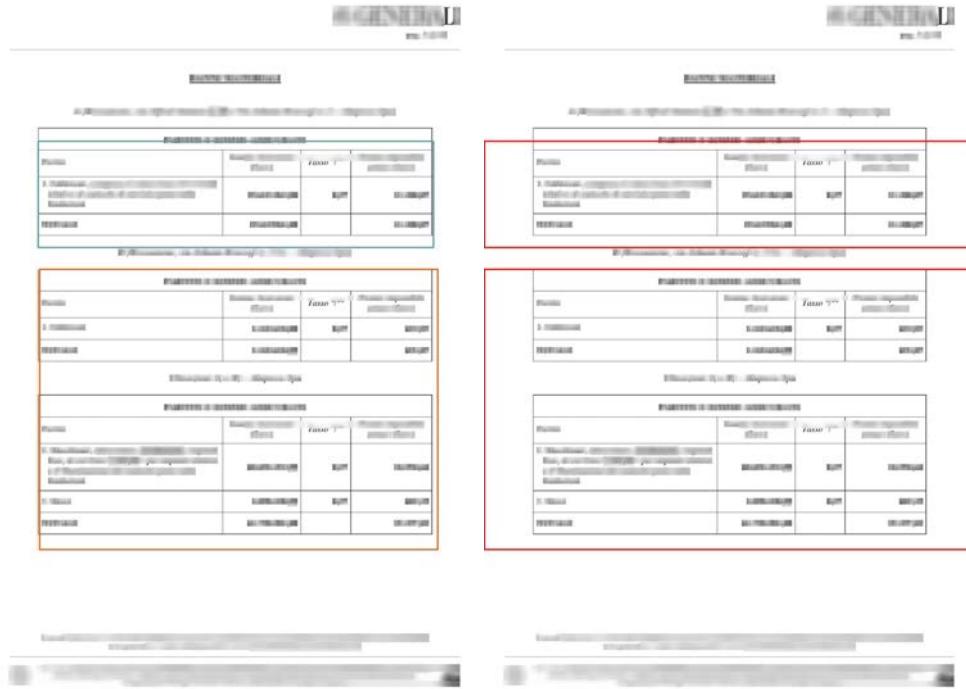


Figura A.10: Test 1, configurazione 2

Configurazione:

```

        }
image_resizer {
    fixes_shape_resizer {
        width: 400
        heigth: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
        }
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
        }
    }
}
use_moving_average: false
}
```

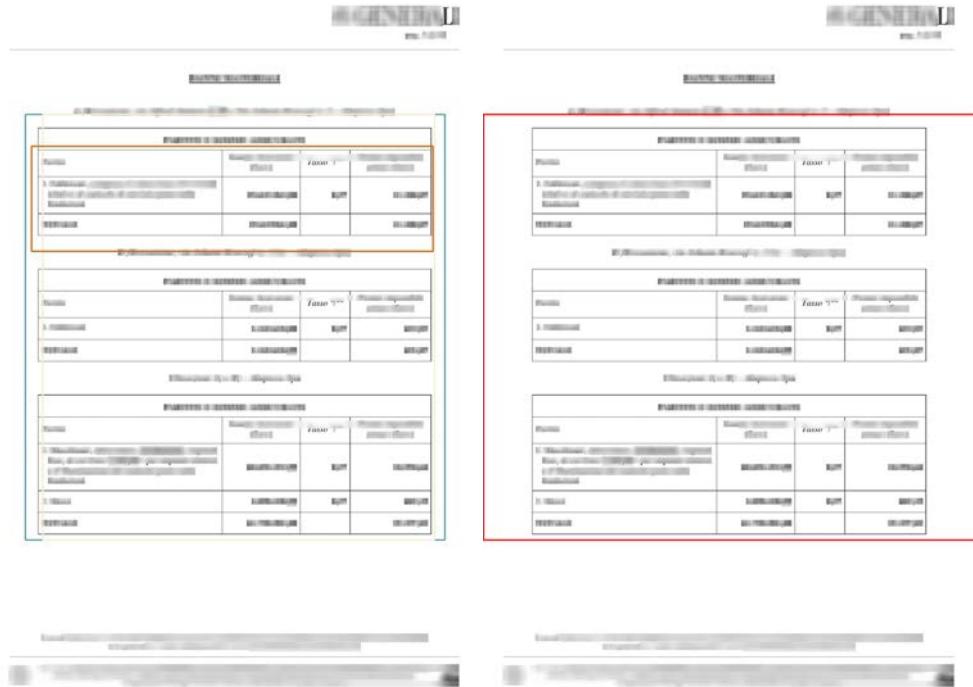
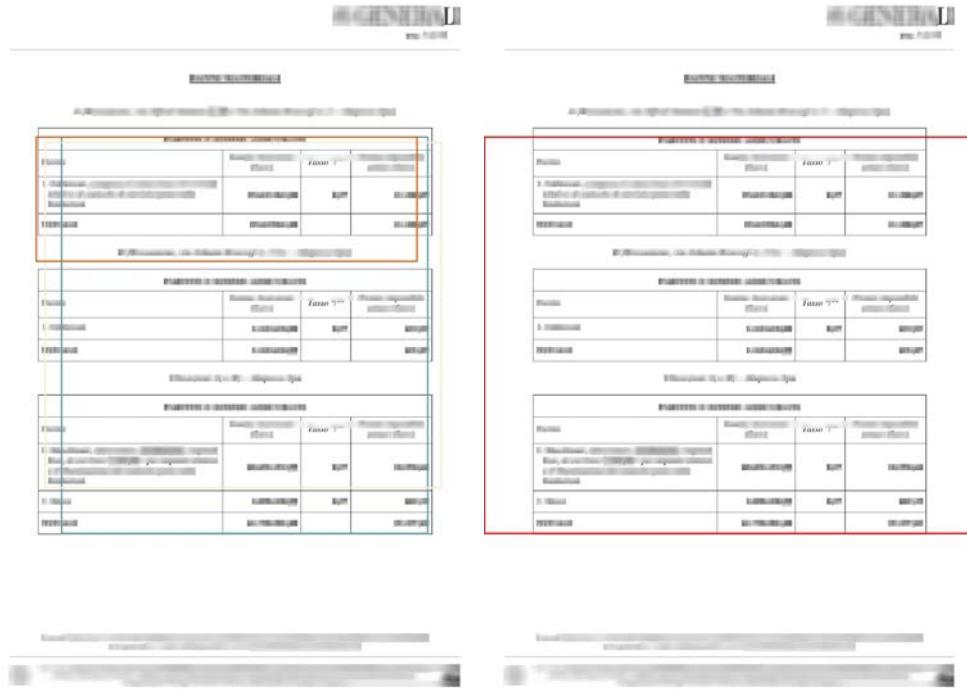


Figura A.11: Test 1, configurazione 3

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
} second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
            ...
        }
        use_moving_average: false
    }
}
}
```

**Figura A.12:** Test 1, configurazione 4

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0008
                schedule {
                    step: 4500
                    learning_rate: .0008
                }
                schedule {
                    step: 7000
                    learning_rate: .0004
                }
                schedule {
                    step: 10000
                    learning_rate: .00008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

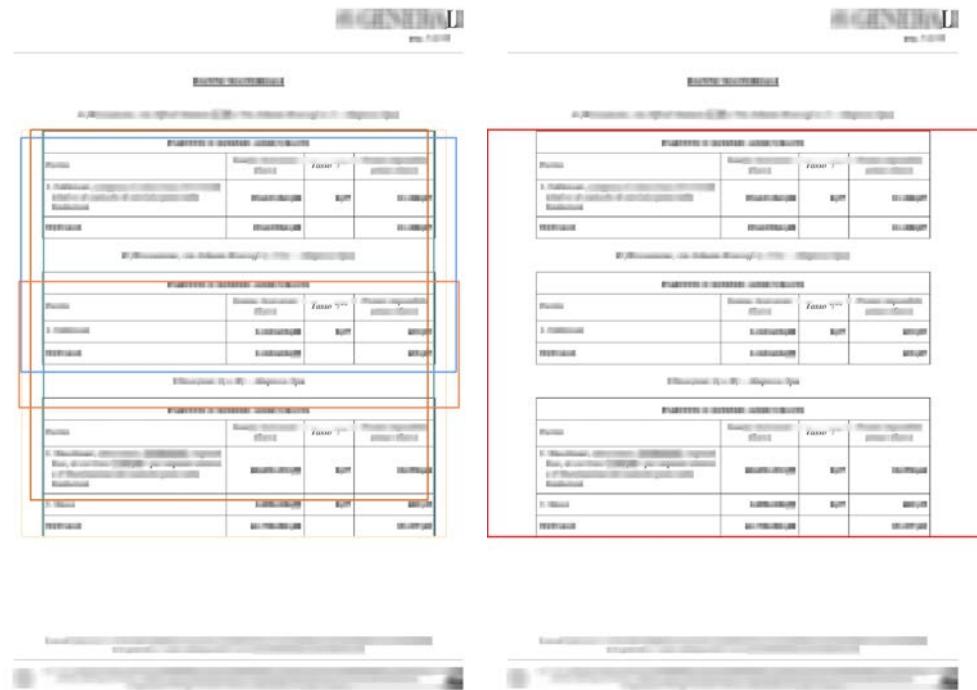


Figura A.13: Test 1, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

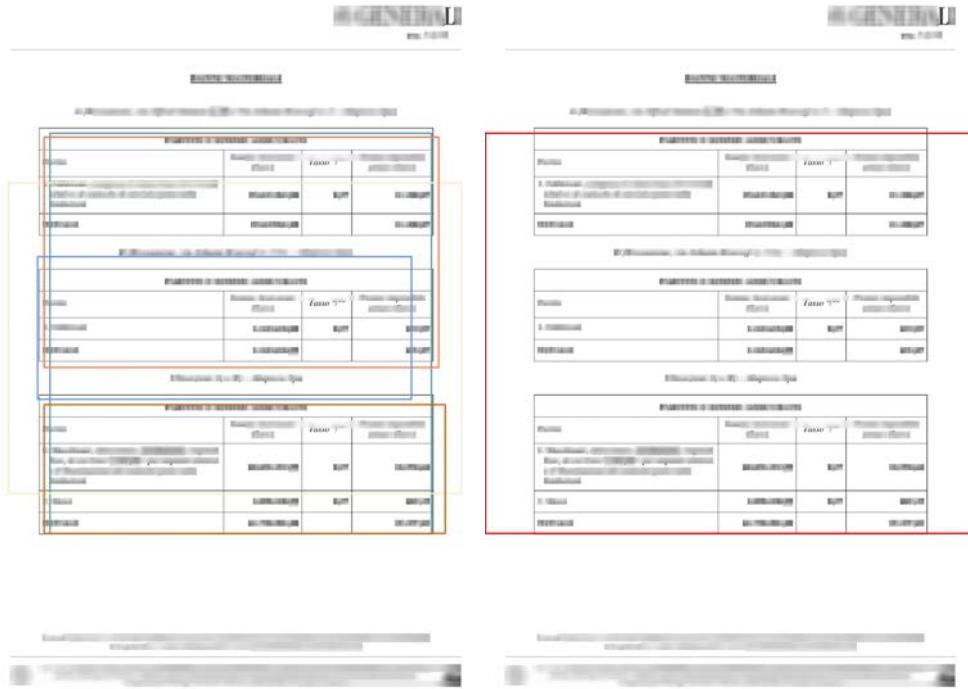


Figura A.14: Test 1, configurazione 6

Configurazione:

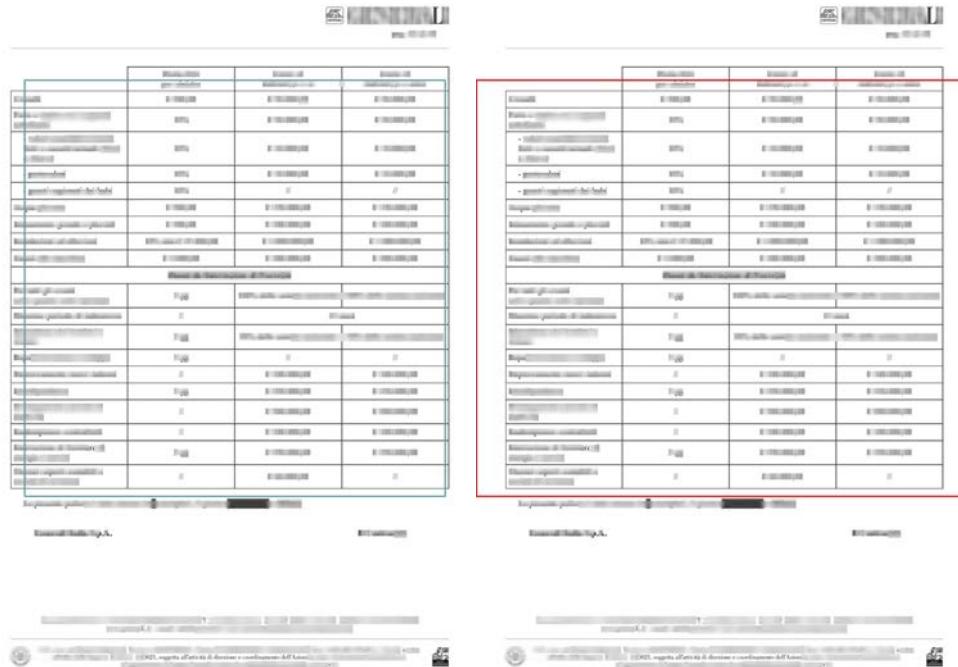
```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

A.1.3 Test 2

Figura A.15: Test 2

**Figura A.16:** Test 2, configurazione 1

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.008
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: .00008
                schedule {
                    step: 4500
                    learning_rate: .00004
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

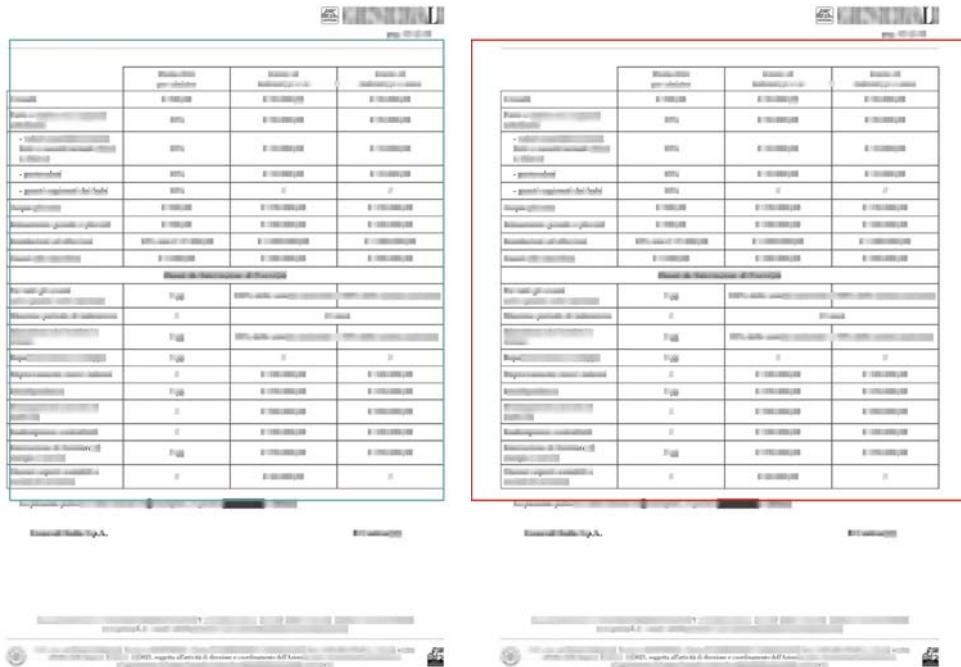


Figura A.17: Test 2, configurazione 2

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
} second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
            ...
        }
        use_moving_average: false
    }
}
}
```

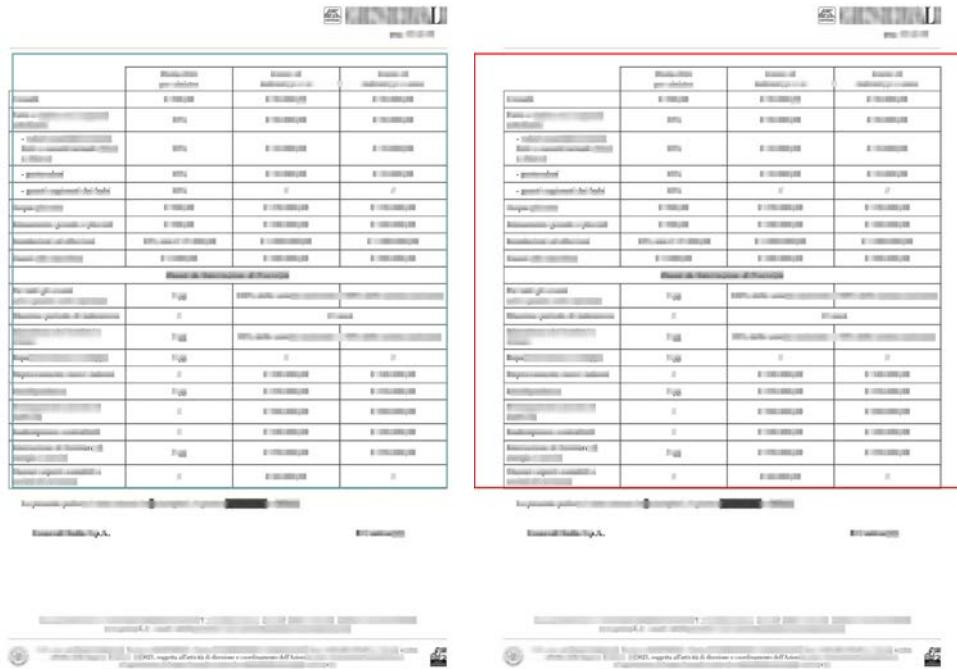


Figura A.18: Test 2, configurazione 3

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
} second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
        }
    }
}
use_moving_average: false
}
```



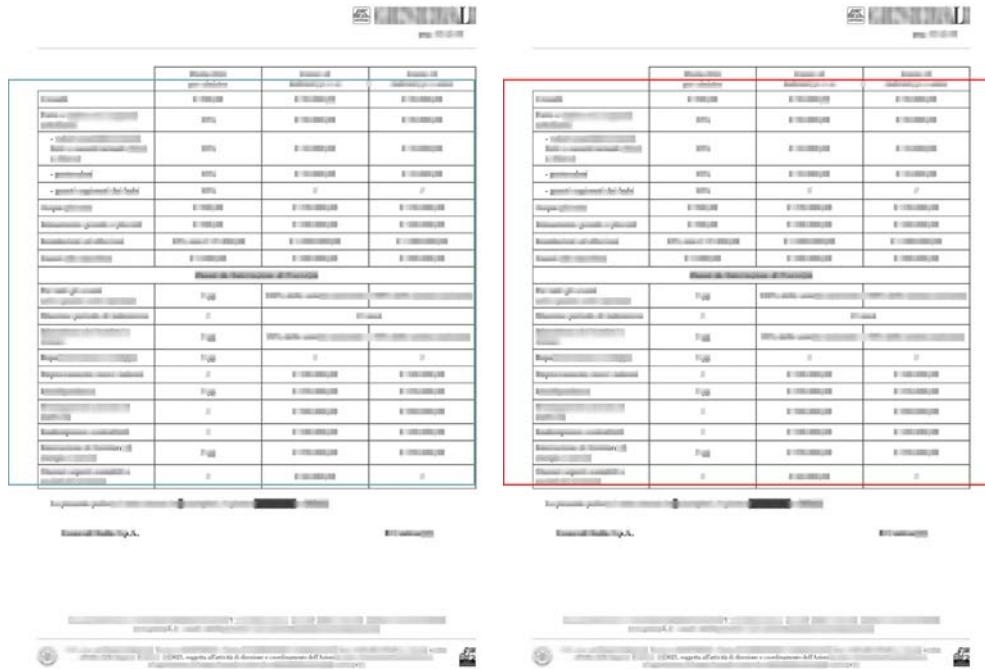
Figura A.19: Test 2, configurazione 4

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0008
                schedule {
                    step: 4500
                    learning_rate: .0008
                }
                schedule {
                    step: 7000
                    learning_rate: .0004
                }
                schedule {
                    step: 10000
                    learning_rate: .00008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

**Figura A.20:** Test 2, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```



Figura A.21: Test 2, configurazione 6

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

A.1.4 Test 3

Figura A.22: Test 3

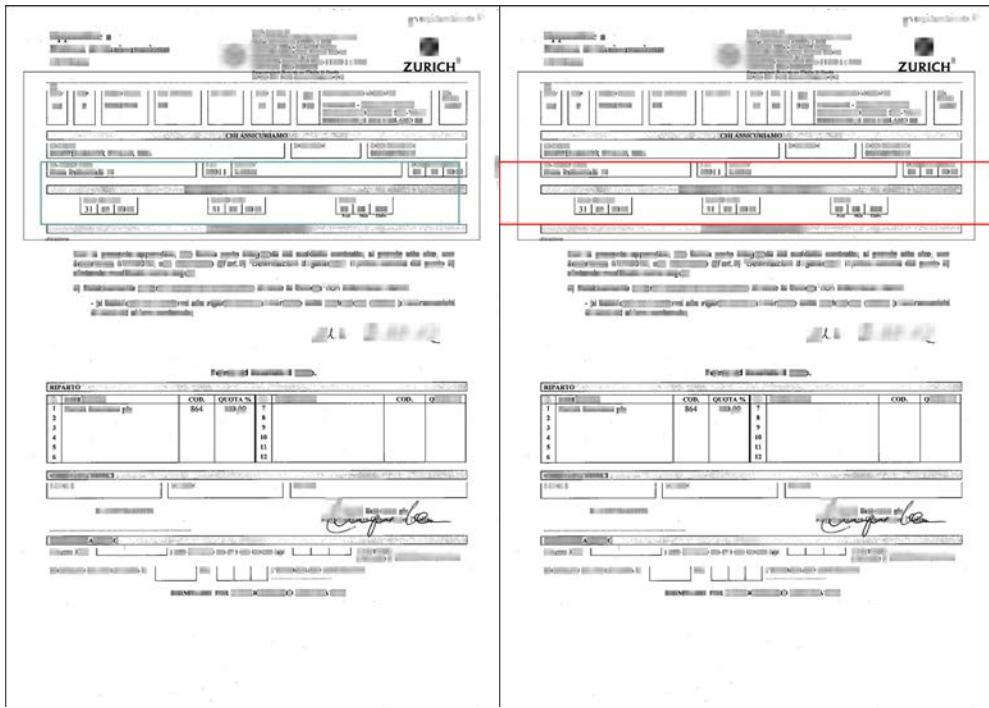


Figura A.23: Test 3, configurazione 1

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.008
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: .00008
                schedule {
                    step: 4500
                    learning_rate: .00004
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

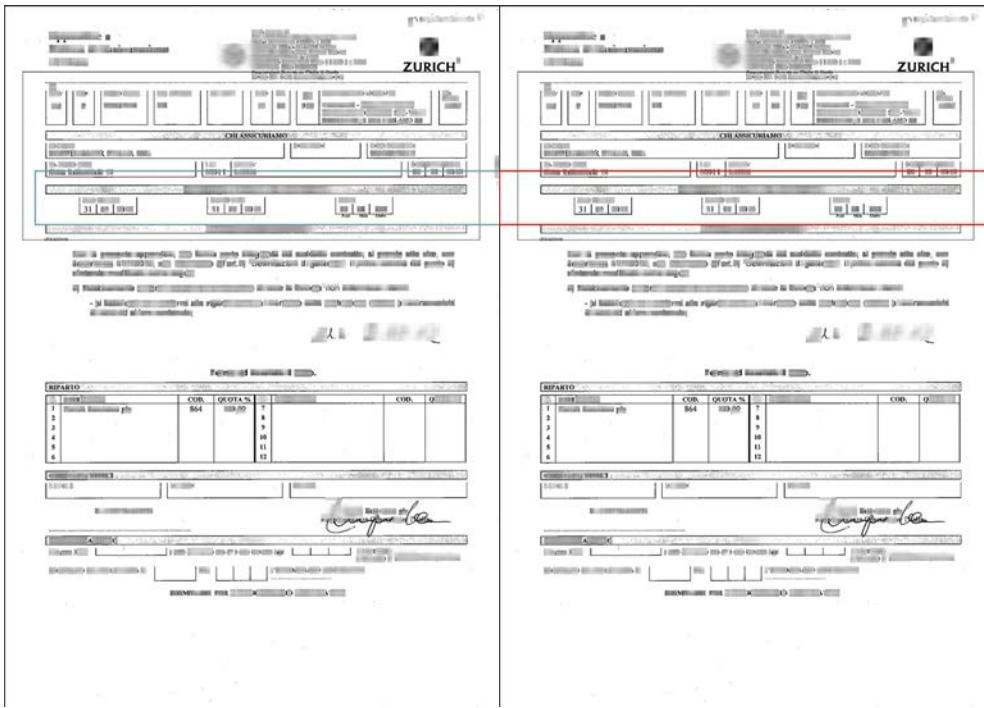


Figura A.24: Test 3, configurazione 2

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}

}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
        }
    }
}
use_moving_average: false
}
}

```

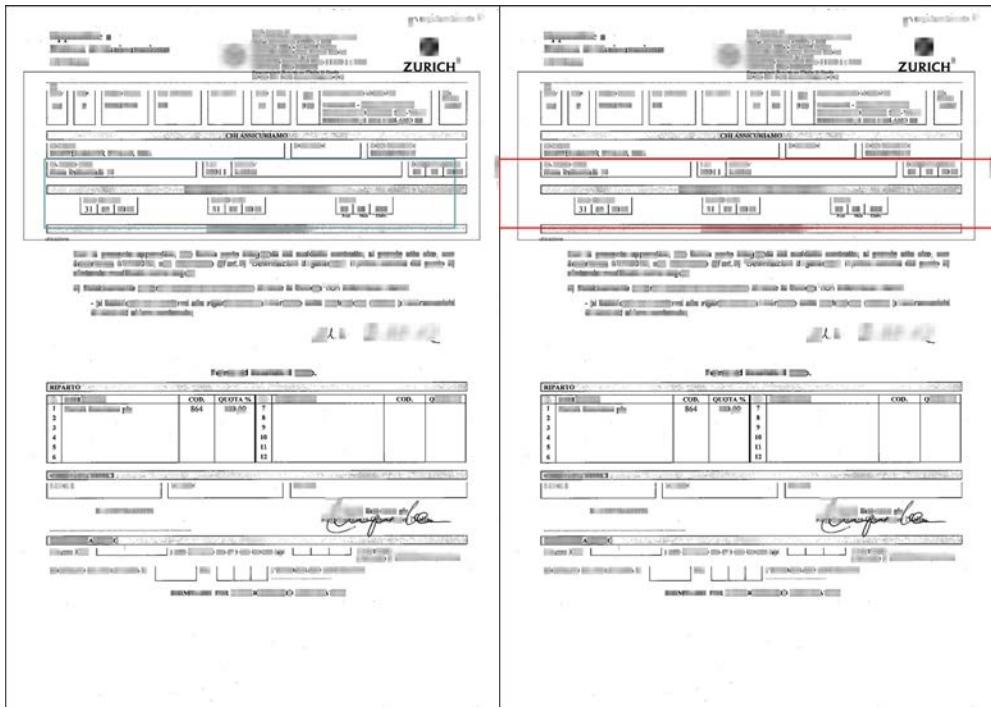


Figura A.25: Test 3, configurazione 3

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
        ...
        use_moving_average: false
    }
}
}
```

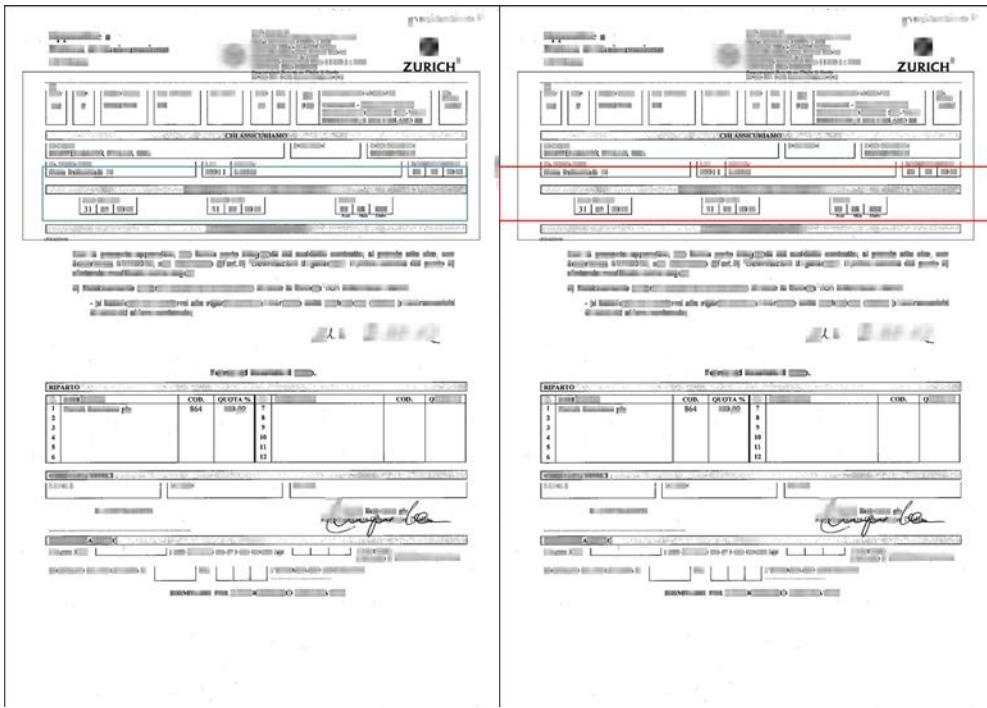


Figura A.26: Test 3, configurazione 4

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0008
                schedule {
                    step: 4500
                    learning_rate: .0008
                }
                schedule {
                    step: 7000
                    learning_rate: .0004
                }
                schedule {
                    step: 10000
                    learning_rate: .00008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

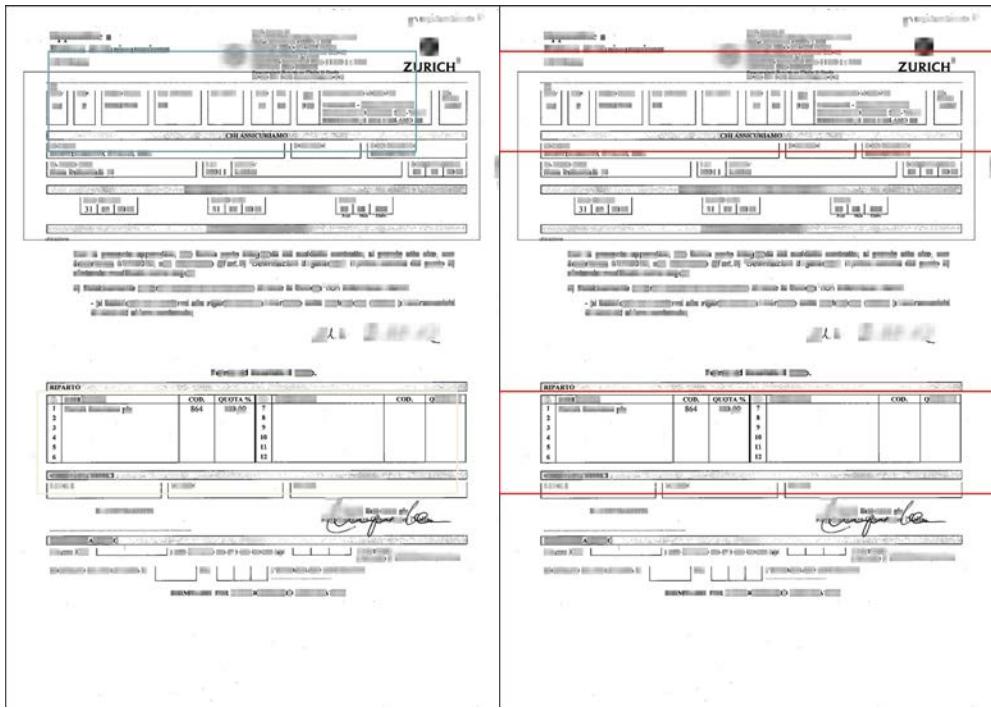


Figura A.27: Test 3, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

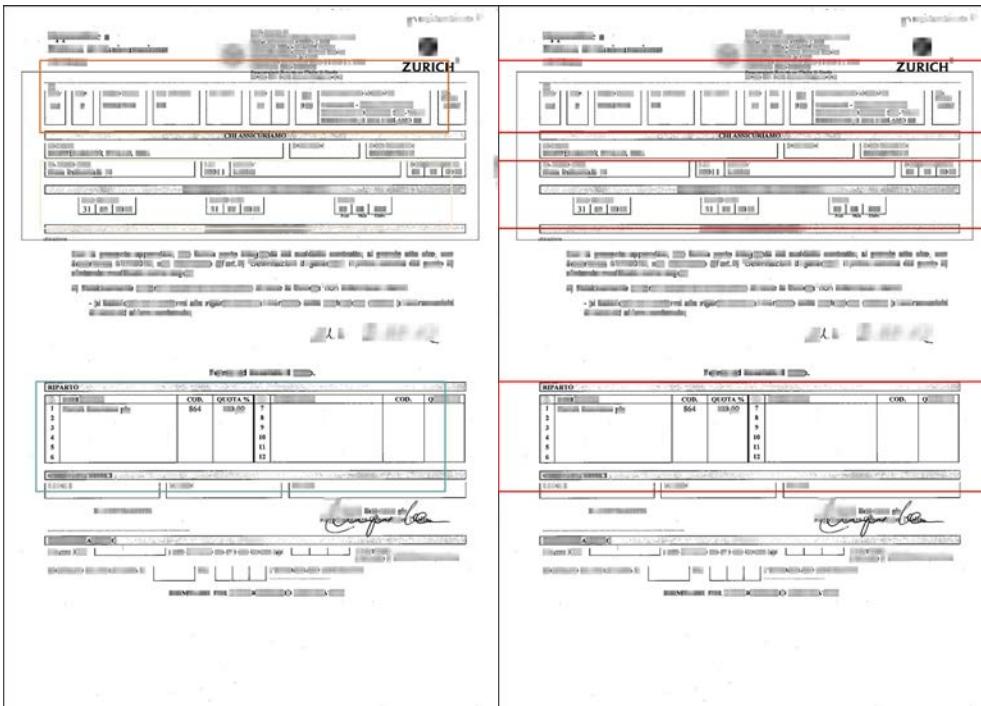


Figura A.28: Test 3, configurazione 6

Configurazione:

```

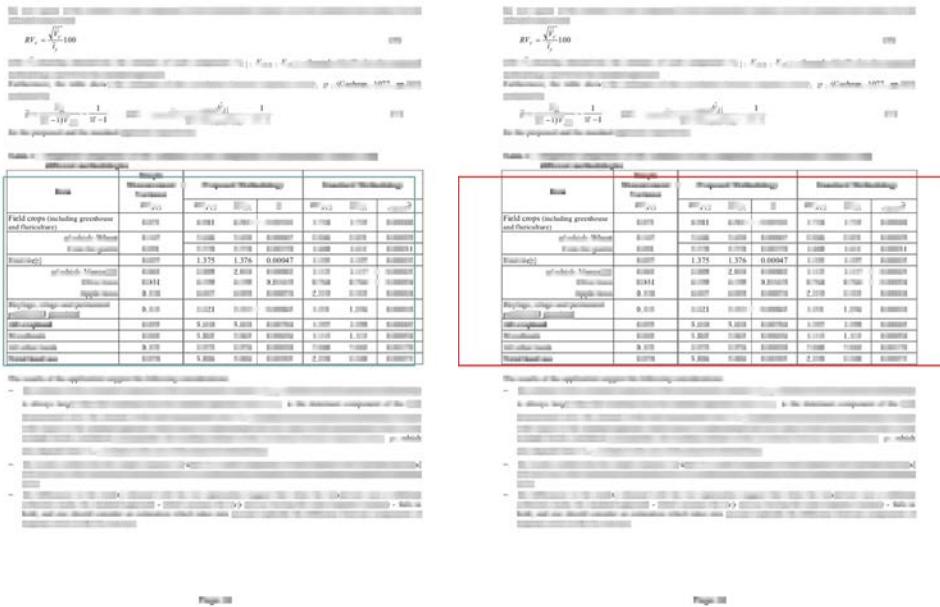
image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
            }
            schedule {
                step: 4500
                learning_rate: .0002
            }
            schedule {
                step: 7000
                learning_rate: .00002
            }
            schedule {
                step: 10000
                learning_rate: .000002
            }
            ...
        }
        momentum_optimizer_value: 0.9
    }
    use_moving_average: false
}

```

A.1.5 Test 4

$$RV_c = \frac{\sqrt{\tilde{V}_c}}{\tilde{t}_y} 100$$

Figura A.29: Test 4



Configurazione:

```

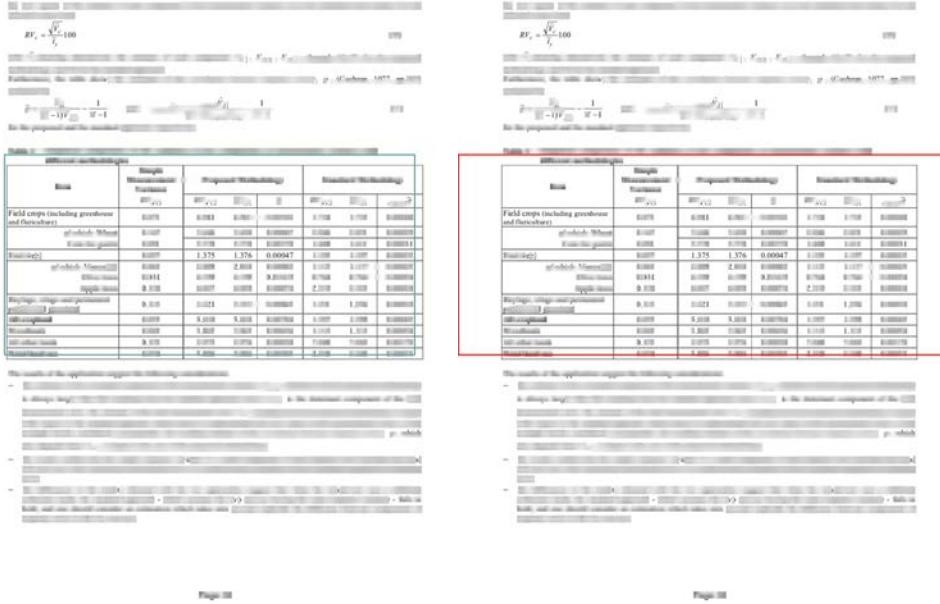
image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.008
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: .00008
                schedule {
                    step: 4500
                    learning_rate: .00004
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

Figura A.31: Test 4, configurazione 2

Configurazione:

```
image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
        }
    ...
    use_moving_average: false
}
}
```

**Figura A.32:** Test 4, configurazione 3

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
} second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
        }
    }
}
use_moving_average: false
}
```

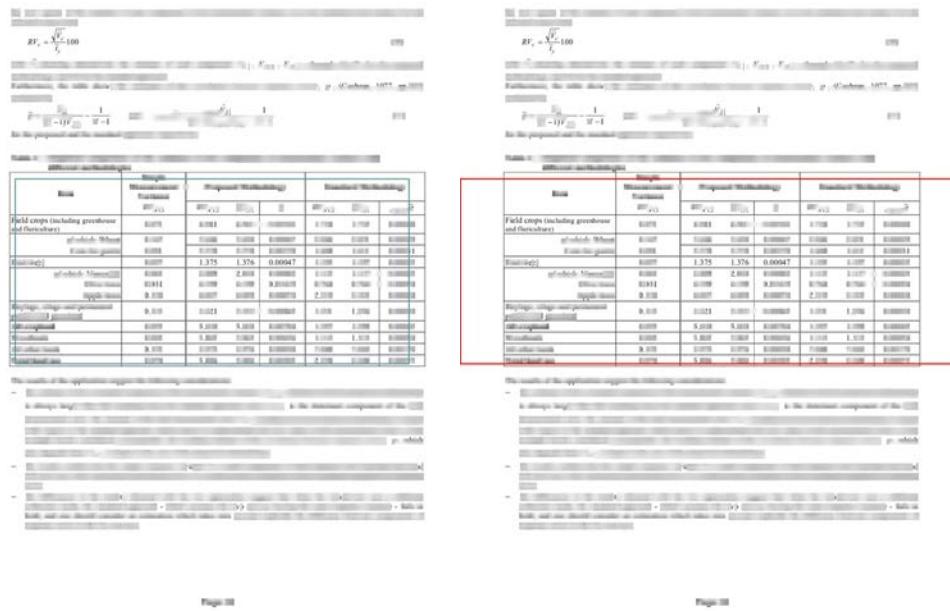


Figura A.33: Test 4, configurazione 4

Configurazione:

```
image_resizer {  
    fixes_shape_resizer {  
        width: 400  
        height: 400  
    }  
}  
first_stage_box_predictor {  
    l2_regularizer {  
        weight: 0.00001  
    }  
    first_stage_nms_iou_threshold: 0.7  
    second_stage_box_predictor {  
        l2_regularizer {  
            weight: 0.00004  
        }  
    }  
    second_stage_post_processing {  
        iou_threshold: 0.6  
    }  
    optimizer {  
        adam_optimizer: {  
            learning_rate: {  
                manual_step_learning_rate {  
                    initial_learning_rate: 0.0008  
                    schedule {  
                        step: 4500  
                        learning_rate: .0008  
                    }  
                    schedule {  
                        step: 7000  
                        learning_rate: .0004  
                    }  
                    schedule {  
                        step: 10000  
                        learning_rate: .00008  
                    }  
                    ...  
                }  
                momentum_optimizer_value: 0.9  
            }  
            use_moving_average: false  
        }  
    }  
}
```

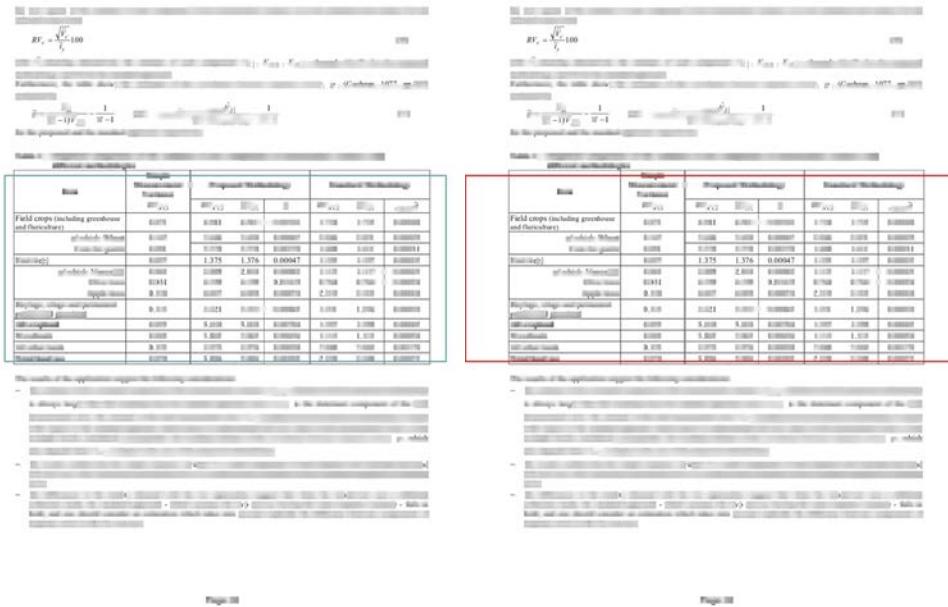


Figura A.34: Test 4, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

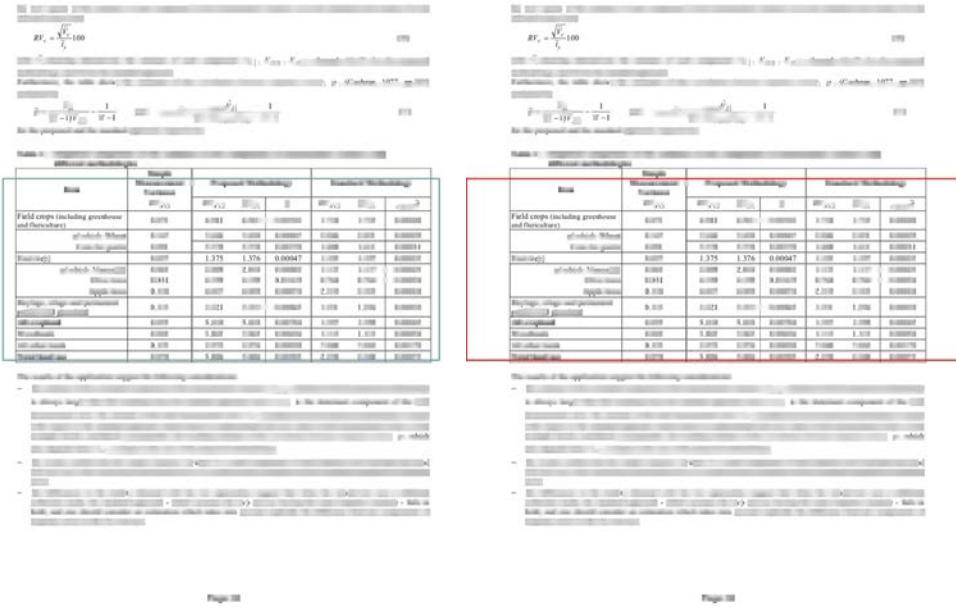


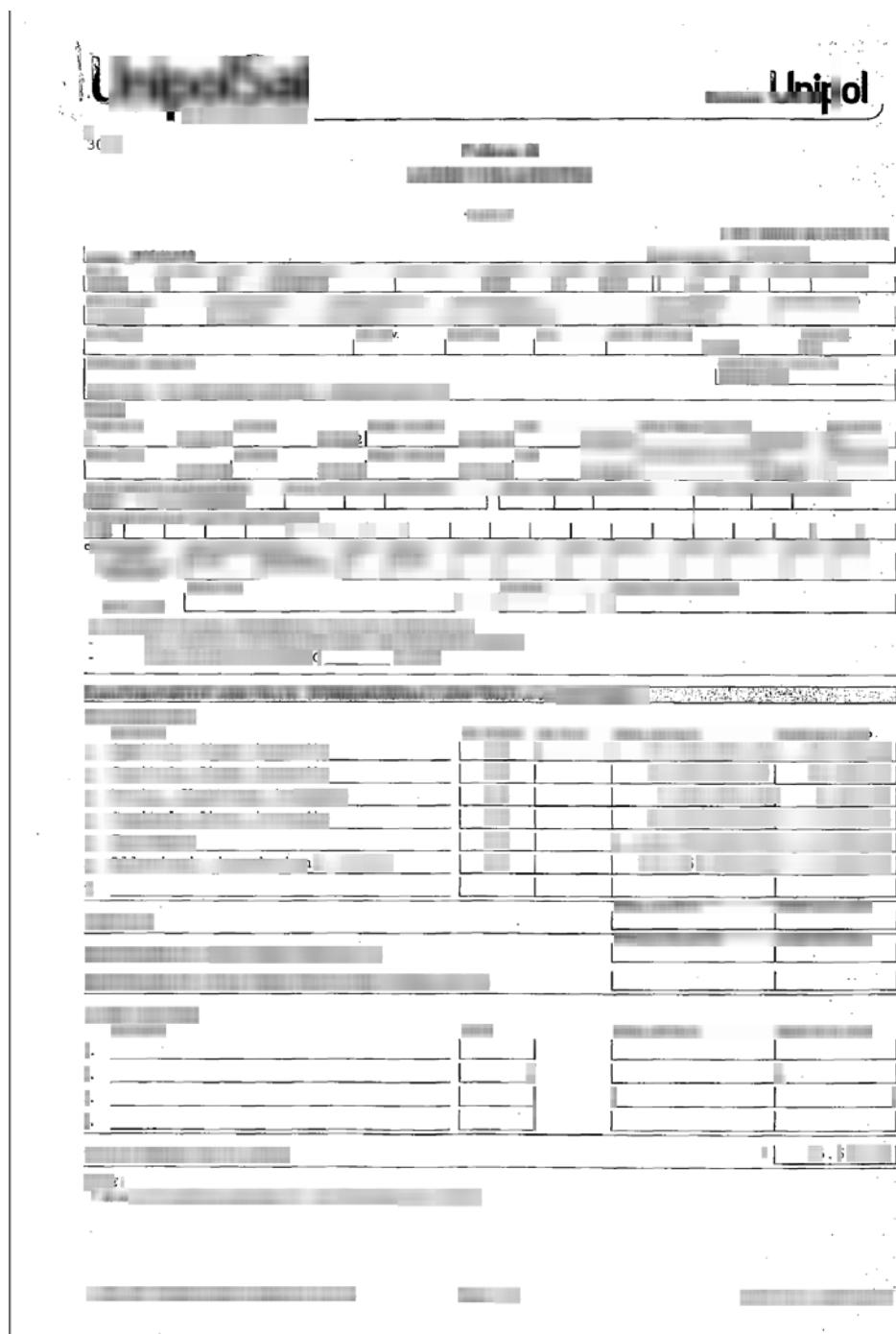
Figura A.35: Test 4, configurazione 6

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

A.1.6 Test 5**Figura A.36:** Test 5

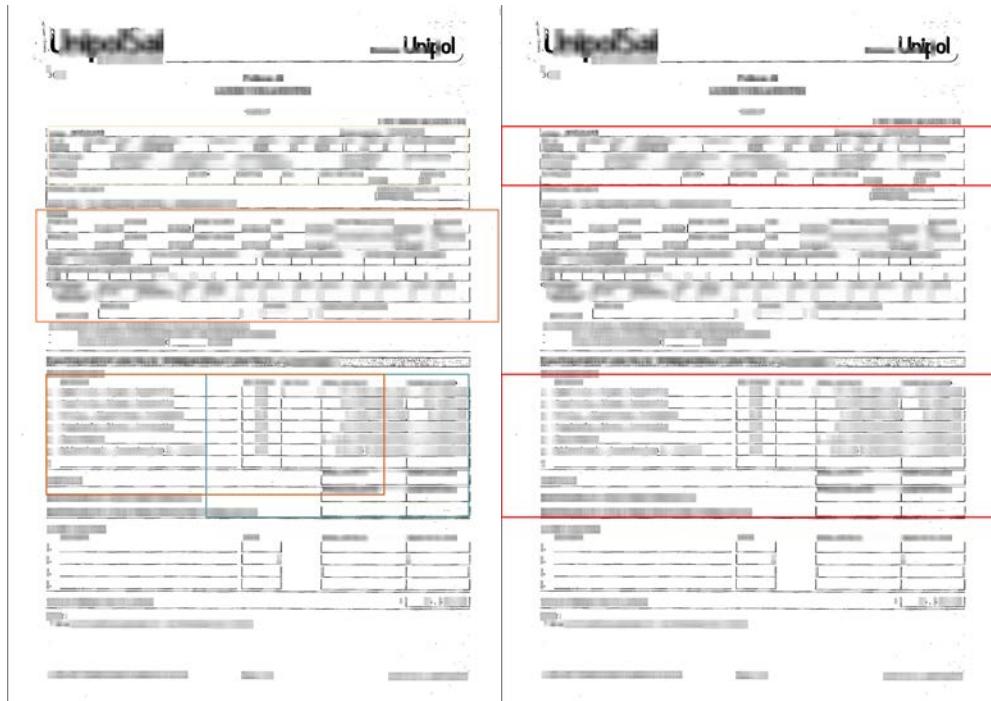


Figura A.37: Test 5, configurazione 1

Configurazione:

```

learning_rate: {
  manual_step_learning_rate {
    initial_learning_rate: .00008
    schedule {
      step: 4500
      learning_rate: .00004
    }
    schedule {
      step: 7000
      learning_rate: .00002
    }
    schedule {
      step: 10000
      learning_rate: .000008
    }
    ...
  }
  momentum_optimizer_value: 0.9
  use_moving_average: false
}

image_resizer {
  fixes_shape_resizer {
    width: 400
    height: 400
  }
}
first_stage_box_predictor {
  l2_regularizer {
    weight: 0.008
  }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
  l2_regularizer {
    weight: 0.004
  }
}
second_stage_post_processing {
  iou_threshold: 0.6
}
optimizer {
  adam_optimizer: {
    ...
  }
}

```

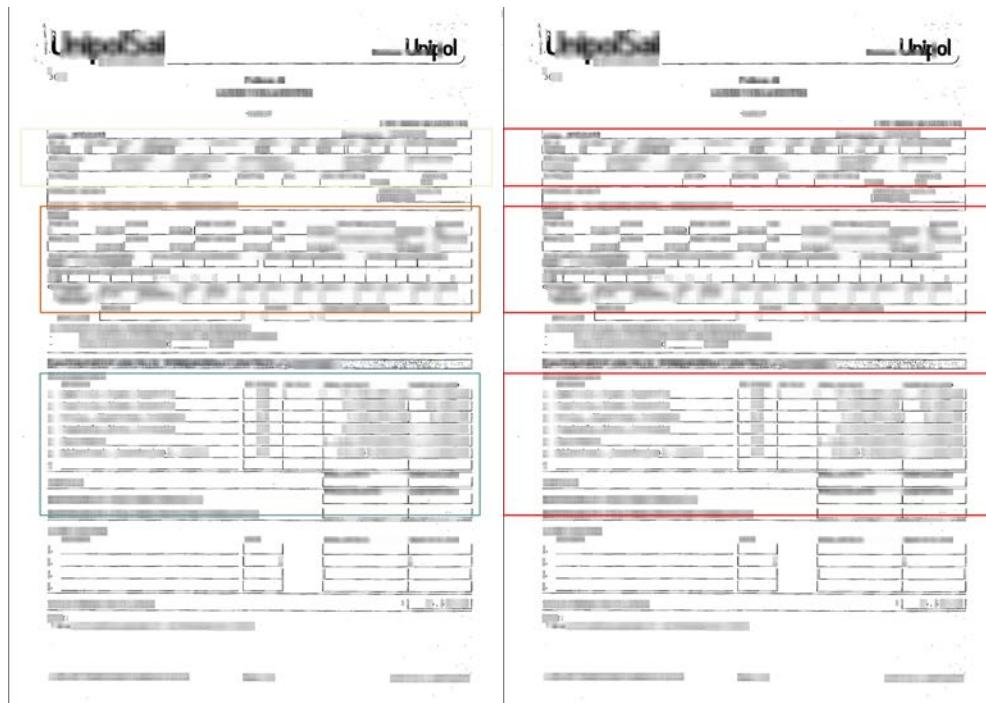


Figura A.38: Test 5, configurazione 2

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 600
                decay_factor: 0.95
            }
        }
    }
}
use_moving_average: false
}
```

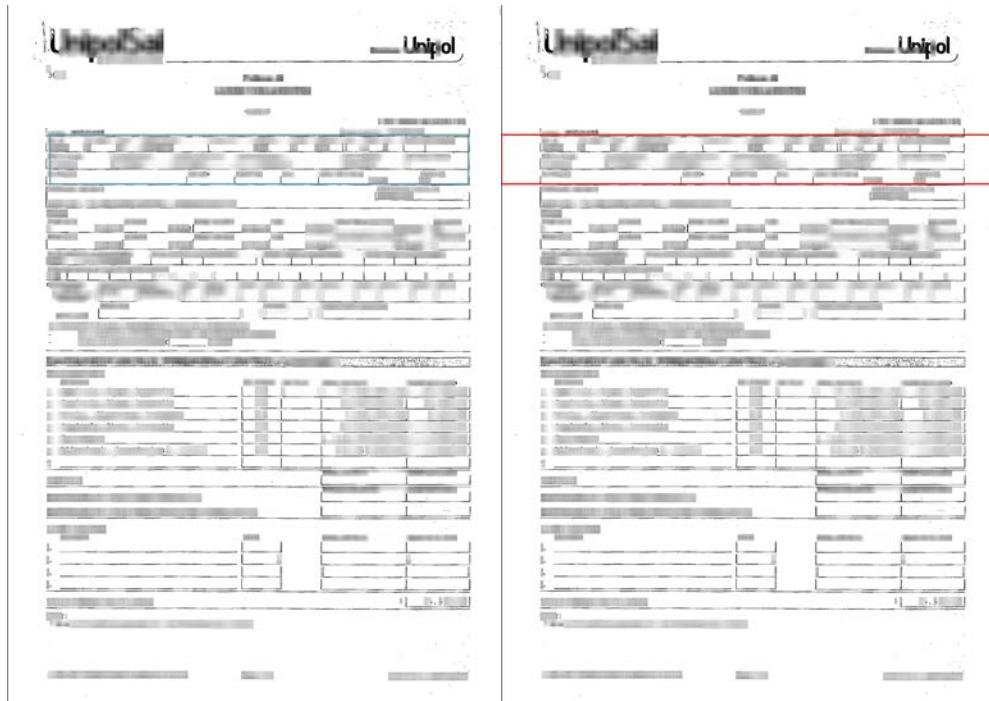


Figura A.39: Test 5, configurazione 3

Configurazione:

```

        }
image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
        }
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            exponential_decay_learning_rate {
                initial_learning_rate: 0.0001
                decay_steps: 450
                decay_factor: 0.9
            }
        }
    ...
    use_moving_average: false
}
}
```

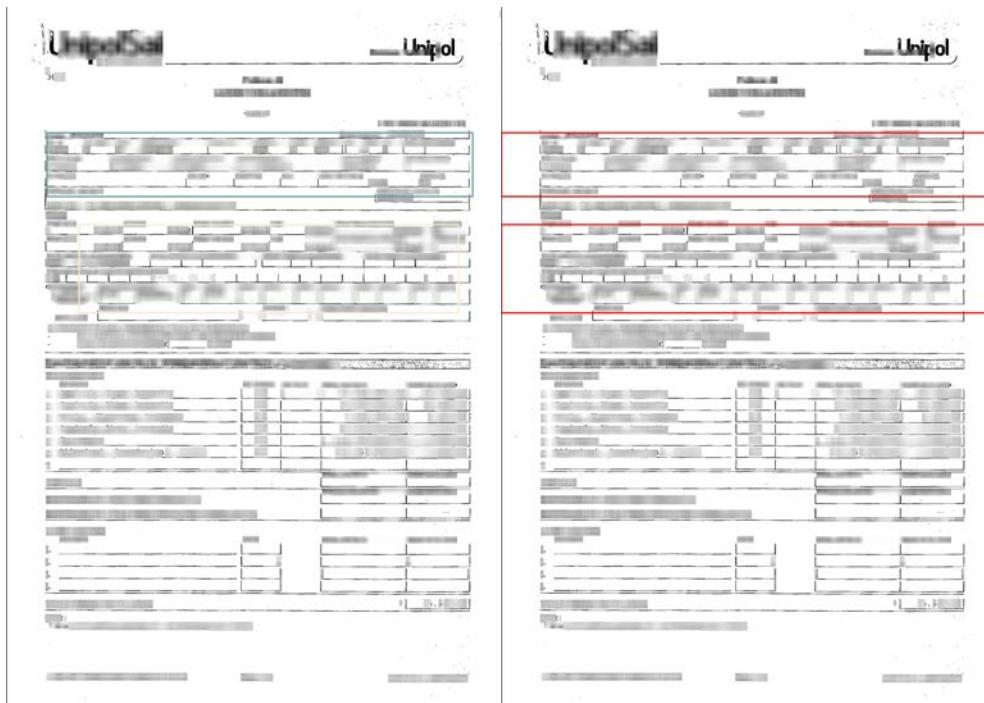


Figura A.40: Test 5, configurazione 4

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.00001
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.00004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0008
                schedule {
                    step: 4500
                    learning_rate: .0008
                }
                schedule {
                    step: 7000
                    learning_rate: .0004
                }
                schedule {
                    step: 10000
                    learning_rate: .00008
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

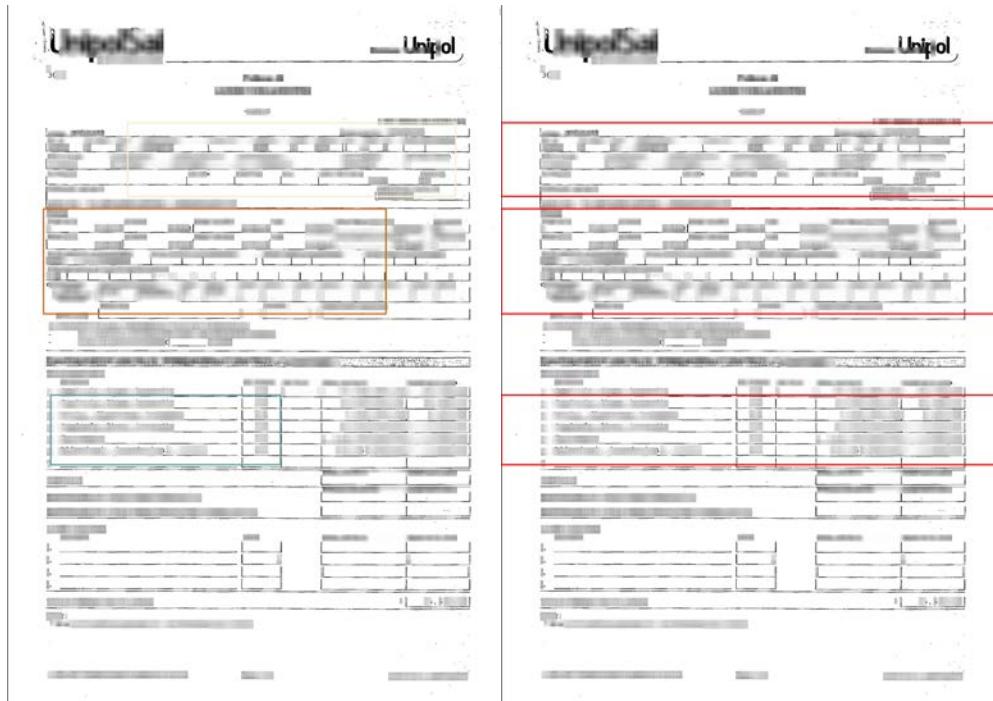


Figura A.41: Test 5, configurazione 5

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

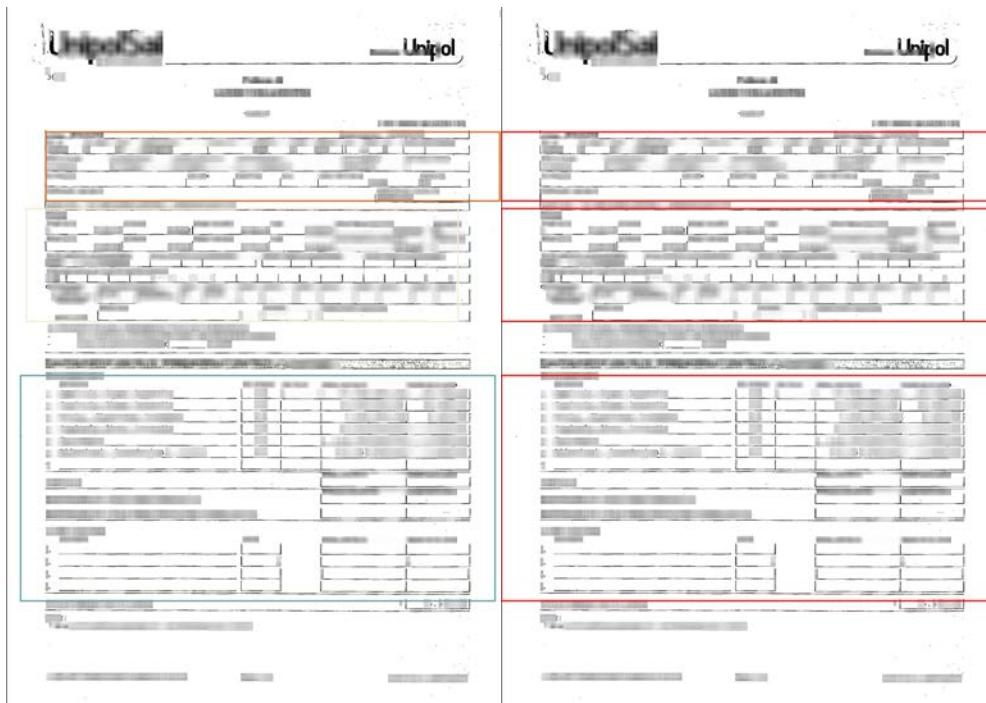


Figura A.42: Test 5, configurazione 6

Configurazione:

```

image_resizer {
    fixes_shape_resizer {
        width: 400
        height: 400
    }
}
first_stage_box_predictor {
    l2_regularizer {
        weight: 0.04
    }
}
first_stage_nms_iou_threshold: 0.7
second_stage_box_predictor {
    l2_regularizer {
        weight: 0.004
    }
}
second_stage_post_processing {
    iou_threshold: 0.6
}
batch_size: 1
optimizer {
    adam_optimizer: {
        learning_rate: {
            manual_step_learning_rate {
                initial_learning_rate: 0.0004
                schedule {
                    step: 4500
                    learning_rate: .0002
                }
                schedule {
                    step: 7000
                    learning_rate: .00002
                }
                schedule {
                    step: 10000
                    learning_rate: .000002
                }
                ...
            }
            momentum_optimizer_value: 0.9
        }
        use_moving_average: false
    }
}

```

A.2 *Crop* da polizze assicurative

In questa sezione vengono mostrati i risultati dell'esecuzione dell'algoritmo su polizze assicurative facenti parte del *dataset* datomi in dotazione. Si potrà notare come in alcuni casi l'inferenza abbia avuto risultati sorprendenti, altri in cui si denotano errori grossolani.

A.2.1 Test 0

Polizza di Assicurazione
All Risks

ZURICH

I UNITÀ*	COMP.	NUMERO POLIZZA		DATA INIZIO POLIZZA	DATA FINO POLIZZA	COD. SRI	SOCI INTERMEDIARIO ASSICURATIVO	INTERIM
A4	P	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]

Premio sino al
31/03/2011

						IMPOSTE	TOTALE EUR	TOTALE
							20.544,30	
		0,00		3.739,15				

N.	COMPAGNIA	COD.	QUOTA %		COD.	QUOTA %
1	[REDACTED]	[REDACTED]	7			
2	[REDACTED]	[REDACTED]	8			
3	[REDACTED]	[REDACTED]	9			
4	[REDACTED]	[REDACTED]	10			
5	[REDACTED]	[REDACTED]	11			
6	[REDACTED]	[REDACTED]	12			

RISK NAME

[REDACTED] [REDACTED]

Figura A.43: Test 0, pagina intera

					TOTALE EUR 20.544,30	TOTALE
	16.805,15	0,00			20.544,30	

Figura A.44: Test 0, tabelle rilevate

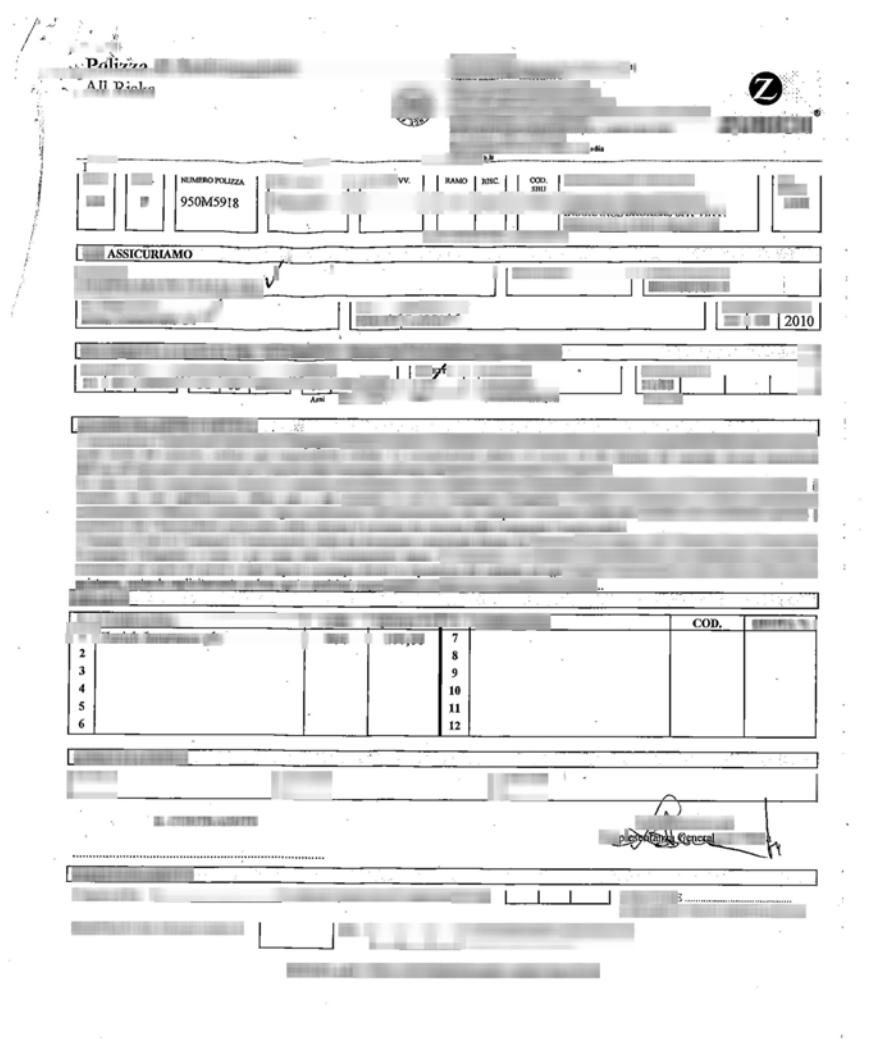


Figura A.45: Test 0, testo rimanente

A.2.2 Test 1

The screenshot shows a software application window with a dark header bar. Below the header, there is a large area of the screen that appears to be heavily redacted or blurred, likely containing sensitive information like names and addresses. At the bottom of the window, there is a table with the following data:

Premesso di assicurato	Somma Assicurata	Tasso	Prezzo Netto
		0,40	2.156,74
		0,40	2.943,68
		0,40	1.601,40
		0,40	3.969,14
		0,40	3.000,00
		0,40	2.734,19
		0,40	400,00
			16.805,15

Figura A.46: Test 1, pagina intera

N.	Partite	Sezioni	Somma Assicurata	Tasso%	Premio Netto
1	[REDACTED] Incluso compres impianti fissi non compres alla partita			0,40	2.156,74
2	[REDACTED]			0,40	2.943,68
3	[REDACTED] compres impianti fissi			0,40	1.601,40
4	[REDACTED]			0,40	3.969,14
5	[REDACTED] in forma fissa			0,40	3.000,00
6	[REDACTED] appurato		5	0,40	2.734,19
7	[REDACTED]			0,40	400,00
TOTALI			42.012,882,00		16.805,16

Figura A.47: Test 1, tabelle rilevate

Figura A.48: Test 1, testo rimanente

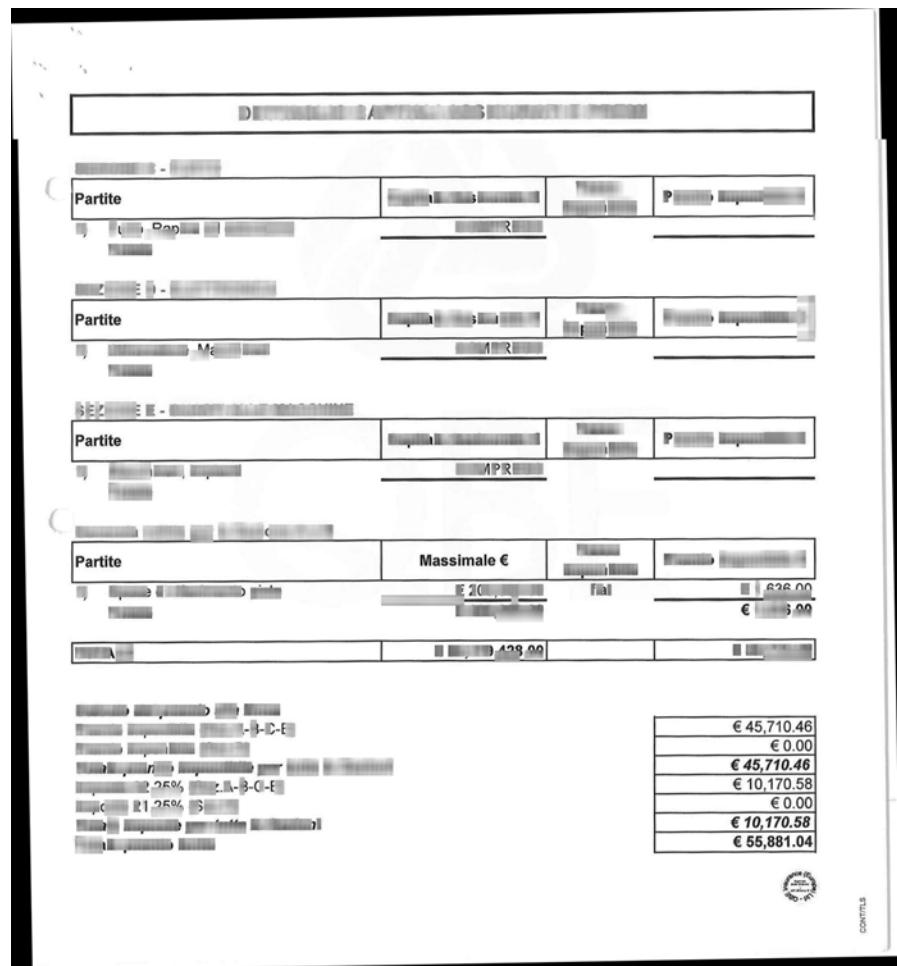
A.2.3 Test 2

Figura A.49: Test 2, pagina intera



The image shows two horizontal tables, each consisting of four columns. The first column is labeled 'Partite' (Parties). The second column contains several lines of text, which are heavily redacted. The third column is labeled 'Tasse' (Taxes) and the fourth column contains numerical values. The first table has a total value of € 3. The second table has a total value of € 1.

Figura A.50: Test 2, tabelle rilevate



The image shows a document with several tables. At the top, there is a large redacted section. Below it, there are three tables, each with four columns. The first table is labeled 'Partite' (Parties), the second is 'Partite' (Parties), and the third is 'SEZIONE E -' (Section E -). The fourth table is labeled 'Partite' (Parties) and 'Massimale €' (Maximum €). It includes a breakdown of amounts: € 20,000.00, Iva (VAT) € 636.00, and a total of € 20,636.00. Below these is a table with several rows of redacted text and a final row showing totals: € 45,710.46, € 0.00, € 45,710.46, € 10,170.58, € 0.00, € 10,170.58, and a final total of € 55,881.04. A small circular logo is at the bottom right, and the word 'continua' is written vertically on the right edge.

Figura A.51: Test 2, testo rimanente

A.2.4 Test 3

L	LINE	R.N.
L		R.N. D.P.
L		12.500,00
F		1.000,00
C		500,00
		500,00
Allagan		500,00
Terremoto	50% delle Somme Assicurate	5.000,00
		5.000,00
Sovraccarico Neve	50% delle Somme Assicurate scoperto 10% - minimo €	5.000,00
		5.000,00



CONTINUED

Figura A.52: Test 3, pagina intera

L	LIN	
L	000,00	
R	000,00	12.500,00
F	000,00	2.000,00
Absoluto	€ 200.000,00	2.500,00
E	000,00	1.500,00
E	000,00	500,00
E	000,00	5.000,00
Inondazioni	50% delle Somme Assicurate	scoperto 10% - minimo € 5.000,00
Allagamenti	000,00	scoperto 10% - minimo € 5.000,00
Terremoto	000,00	scoperto 10% - minimo € 5.000,00
Franamenti e Smottamenti del terreno	000,00	10% - minimo € 5.000,00
Sovraccarico Neve	50% delle Somme Assicurate	scoperto 10% - minimo € 5.000,00

Figura A.53: Test 3, tabelle rilevate

Figura A.54: Test 3, testo rimanente

A.2.5 Test 4

The image shows a redacted document from ITAS Mutua. At the top left is the ITAS Mutua logo, which includes a stylized eagle and the text 'ITAS MUTUA'. Above the logo, there is some very small, illegible text. Below the logo is a large, heavily redacted rectangular area containing several lines of text. In the center of this redacted area, the amount '100.000,00' is visible. To the right of this central area, there is another section of redacted text. At the bottom left of the redacted area, the text 'verso la stessa assicurativa nella' is partially visible. To the right of this, the text '50% delle som' is visible. Below this main redacted section, there is a smaller, separate redacted area consisting of several horizontal lines. At the very bottom left, the word 'GARANZIA' is printed in bold capital letters, followed by a redacted table with four columns and four rows.

Figura A.55: Test 4, pagina intera

Figura A.56: Test 4, tabelle rilevate

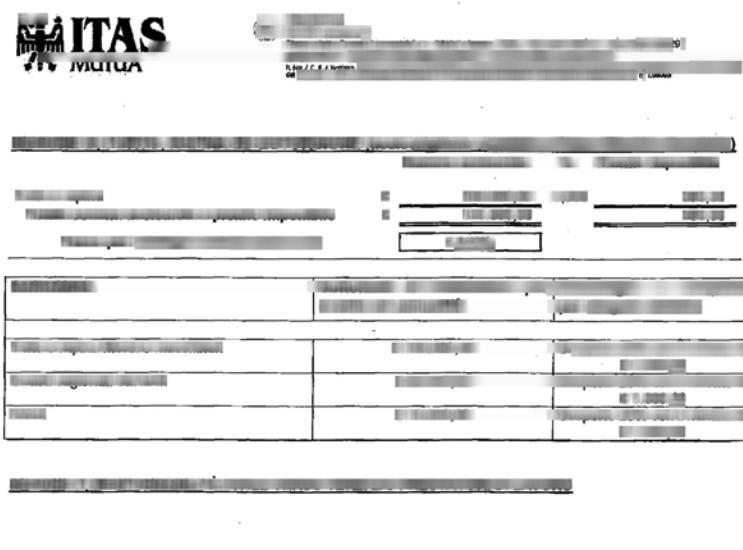


Figura A.57: Test 4, testo rimanente

A.2.6 Test 5

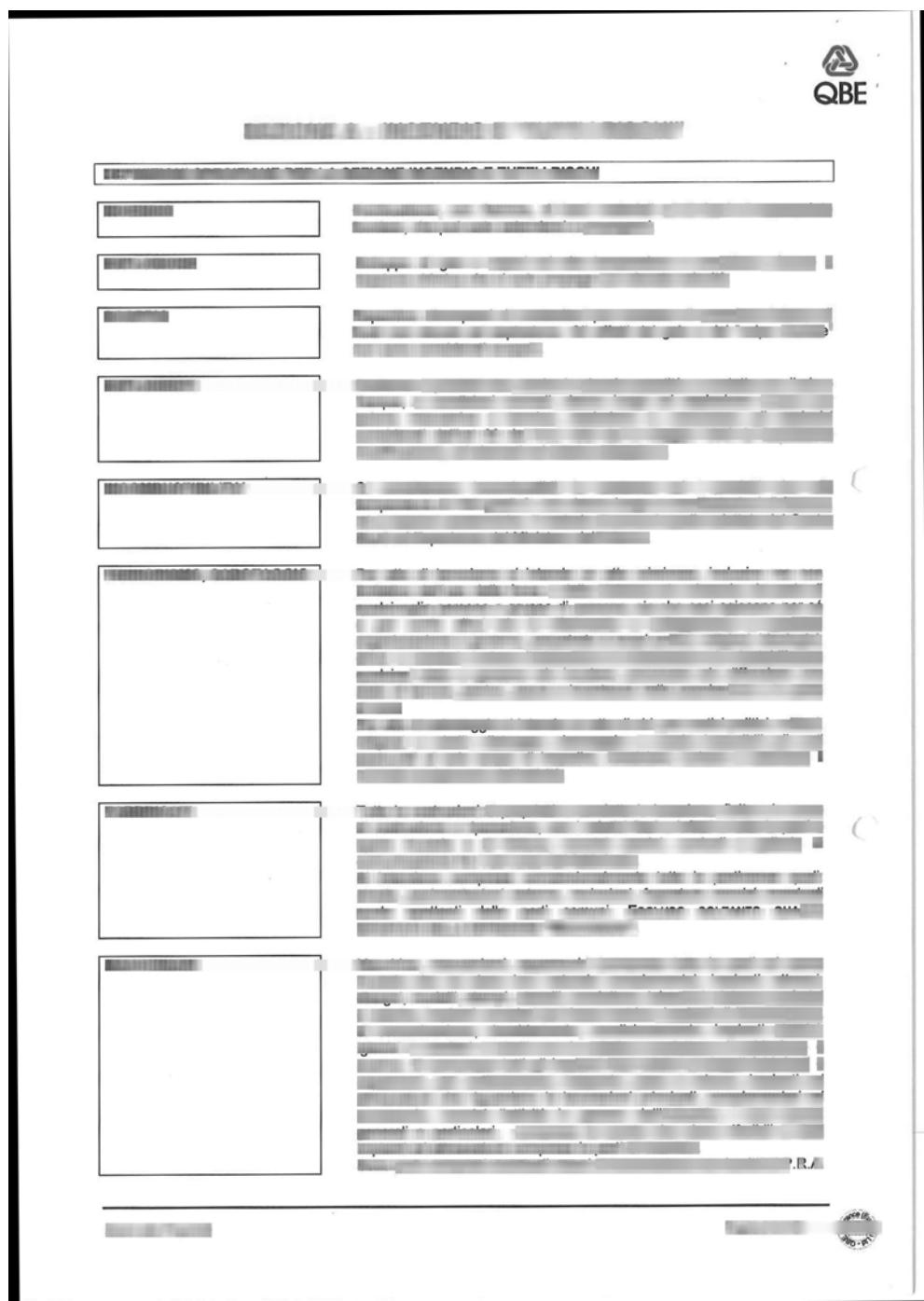


Figura A.58: Test 5, pagina intera

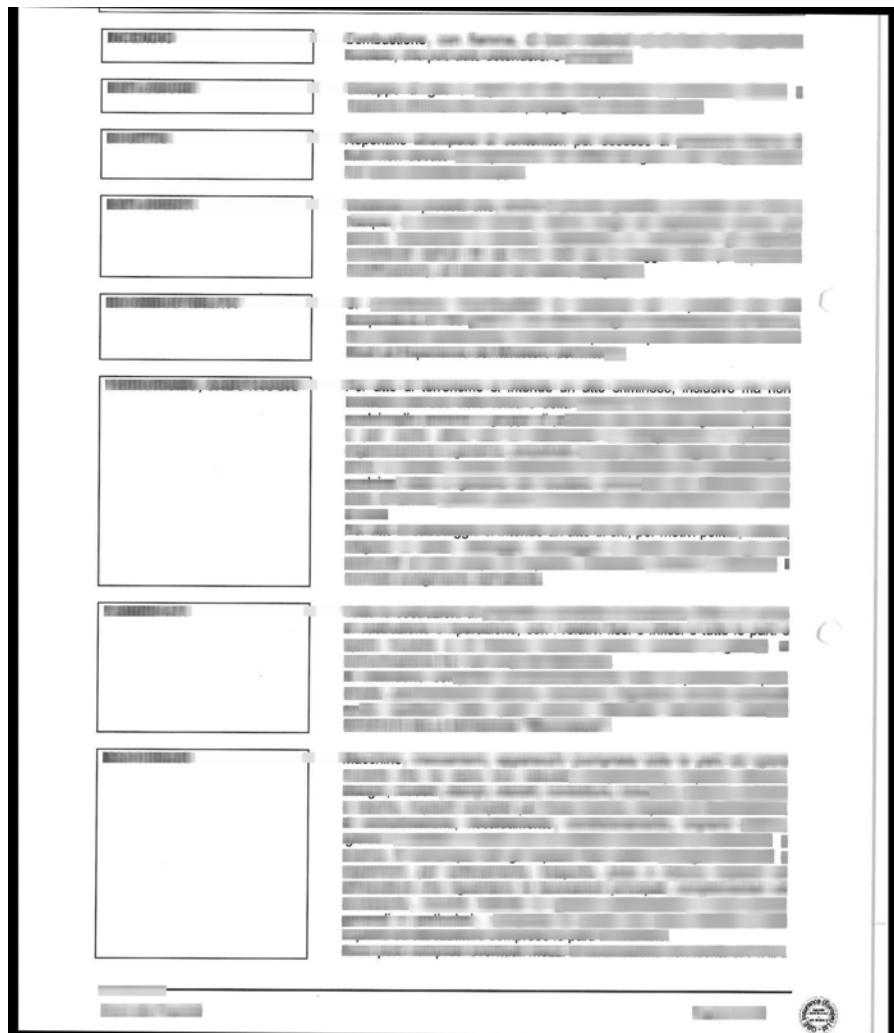


Figura A.59: Test 5, tabelle rilevate

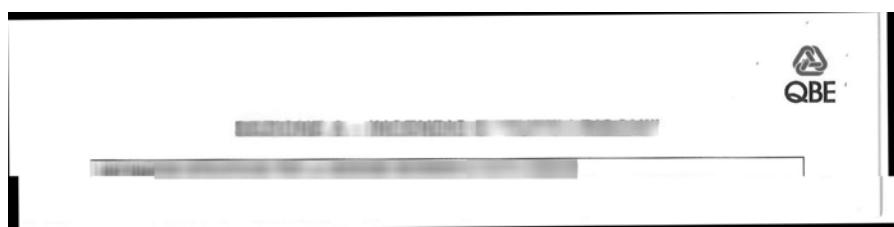


Figura A.60: Test 5, testo rimanente

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [99](#)

Amazon Web Services Amazon Web Services offre servizi di cloud computing affidabili, scalabili ed economici. L'account è gratuito e si pagano solo i servizi usati.. [99](#)

batch size numero di input da processare in una valutazione. Se pari a al numero di elementi in input equivale ad eseguire una valutazione sull'intero set di dati per *epoch*. Se lo si diminuisce si ha la possibilità di validare solo una parte dei dati, con i quali eseguire delle stime, quindi procedere ad una validazione di un altro *set* di dati.. [24](#)

canale In generale, quando si parla di canali nelle reti neurali ci si riferisce spesso alla terza dimensione di una matrice cubica, ovvero alla "profondità" che essa ha. Pensando ad una immagine a tre canali (uno per ogni colore primario), ogni canale corrisponde ad una matrice quadrata della dimensione dell'immagine stessa.. [15](#)

classe Una classe nelle reti neurali corrisponde ad una caratteristica che si vuole ricercare all'interno di un set di dati. Pensando all'*object detection*, in un'immagine ogni classe corrisponde ad un determinato oggetto che vogliamo individuare: un lampione, una ruota, una persona, ecc.. [16](#)

CNN una *Convolutional Neural Network* è una rete neurale a strati di convoluzione.. [99](#)

convoluzione la convoluzione è un'operazione matematica che descrive una regola per miscelare due funzioni oppure due pezzi distinti di un'informazione. Nelle reti neurali per l'*object detection*, maschere di convoluzione servono per ricavare informazioni diffuse all'interno di immagini, come ad esempio i bordi.. [13](#)

data augmentation talvolta può essere utile generare un *dataset* sintetico a partire dai propri dati. Questo viene generato tramite il *data augmentation* e corrisponde ad alcune modificazioni "standard" che si possono fare alle immagini, come ad

esempio: ritaglio casuale, ingrandimenti, riduzione di dimensione, rotazione, ecc..
[24](#)

funzione di attivazione una funzione di attivazione è una funzione matematica che agisce come una sorta di filtro che decide se il risultato è abbastanza "buono" da poter essere inoltrato al nodo successivo. La più nota e attualmente utilizzata è la *ReLU* che è così composta:

$$\text{ReLU}(x) = \max(x, 0)$$

che quindi prende il massimo tra il numero e 0, escludendo così gli output negativi.
[. 10](#)

generators i generatori sono un tipo di iterabile, come le liste e le tuple. Ma a differenza delle liste, i generatori non permettono l'indicizzazione con indici arbitrari e permettono di dichiarare una funzione che si comporti come un iteratore, utilizzandola poi in un ciclo for.. [34](#)

hard negative mining è una tecnica che consiste nel creare forzatamente degli esempi negativi all'interno di un *dataset*. Facciamo un esempio: in un *dataset* di volti, per fare l'*hard negative mining* è necessario prendere porzioni di immagine dove non sono presenti tabelle ed etichettarli come "volto non presente". Questo abbassa notevolmente i falsi positivi.. [21](#)

Integrated Development Environment in informatica, è un software che, in fase di programmazione, aiuta i programmatore nello sviluppo del codice sorgente di un programma.. [99](#)

IoU L'*Intersection over Union* è una metrica di valutazione usata per misurare l'accuracy di una *object detection* in un particolare dataset. Si tratta di una semplice metrica di valutazione: un qualsiasi algoritmo che offre delle scatole di predizione può essere valutato usando l'IoU.

Per applicare l'IoU bisogna avere sia le etichette originali dei dati, che le etichette ricavate dall'allenamento. Dopodiché il calcolo è semplice, poiché è l'area sovrapposta diviso l'area dell'unione risultante delle due etichette. Ovviamente più alto è il rapporto più l'etichetta ricavata dall'allenamento sarà precisa.. [99](#)

learning rate tasso di apprendimento. In una rete neurale esso rappresenta la "velocità" con cui una rete può imparare. È il parametro principale da regolare quando si vuole allenare una rete neurale.. [24](#)

Long Short-Term Memory la LSTM è una tipologia di *recurrent neural network* che è provvista di un *forget gate*, che è utile per dimenticare, dopo un certo lasso di tempo, alcune *features* imparate in precedenza. Può essere utile per imparare da dati dove possono esserci presenti dei vuoti di lunghezza non conosciuta.. [99](#)

max-pooling il *pooling* è un'operazione matematica che prende in ingresso una serie di input e li riduce ad un singolo valore. Quindi, in particolare, il *max-pooling* prende come input una matrice di dimensione variabile e ne restituisce il valore massimo.. [12](#)

non-maximum suppression è una tecnica per eliminare punti che non corrispondono a zone dove esistono bordi rilevanti.. [21](#)

Optical Character Recognition i programmi che si occupano di fare OCR sono dedicati al rilevamento dei caratteri contenuti in un documento e al loro trasferimento in testo digitale leggibile da una macchina.. [99](#)

overfitting in statistica e in informatica, si parla di *overfitting* (in italiano: eccessivo adattamento) quando un modello statistico molto complesso si adatta ai dati osservati (il campione) perché ha un numero eccessivo di parametri rispetto al numero di osservazioni.. [27](#)

POD Page Object Detection è uno strumento di *computer vision* che si occupa di trovare degli oggetti caratteristici all'interno di documenti, come grafici, tabelle e immagini. [99](#)

R-CNN una *Region-based Convolutional Neural Network* è una rete neurale a strati di convoluzione che ha, come input, delle regioni "proposte" da parte di un altro algoritmo.. [99](#)

regressione lineare un problema di regressione consiste nel prevedere il valore di una variabile numerica in base ai valori di uno o più variabili predittive, che possono essere numerici o categorici. Nel caso citato, ci si riferisce ad un metodo matematico di calcolo di regressione che è lineare.. [18](#)

REpresentational State Transfer è uno stile architetturale, ovvero un'astrazione degli elementi di un'architettura all'interno di un sistema distribuito.. [99](#)

set parola che occorre spesso nel testo, indica un insieme di dati e, nel contesto di questa tesi, in particolare un insieme di dati tra di loro concordi, che abbiano al loro interno una etichetta o un qualsiasi tipo di informazione inherente al contenuto dei dati stessi. [10](#)

softmax una funzione *softmax* è una generalizzazione di una funzione logistica che comprime un vettore n -dimensionale di valori reali arbitrari in un vettore n -dimensionale di valori compresi tra $(0, 1)$ la cui somma è 1.. [18](#)

SVM sono anche chiamate macchine *kernel* e sono delle metodologie di apprendimento supervisionato per la regressione e la classificazione di *pattern*, sviluppati negli anni '90 da Vladimir Vapnik e il suo team presso il laboratorio Bell della AT&T. [99](#)

vanishing gradient fenomeno che accade spesso nelle reti molto profonde e che non hanno degli strumenti di *backpropagation* adeguati, consiste nel fatto che il gradiente emesso dalla funzione errore decresce esponenzialmente non appena esso viene propagato all'indietro ai precedenti *layers*. In pratica, nel tempo in cui l'errore viaggiava "all'indietro" nella rete verso i *layers* antecedenti, esso diventa così piccolo che l'apprendimento conseguente risulta quasi pari a zero. . [13](#)

zero-padding il *zero-padding* è una tecnica di trasformazione di immagini che consiste nell'aggiungere degli zeri attorno ad una matrice di pixel per aumentarne le dimensioni. Viene spesso usato per far coincidere le dimensioni di matrici diverse quando informazioni vengono passate tra uno strato e l'altro.. [13](#)

Acronimi

API Application Program Interface. 17

AWS Amazon Web Services. 39

CNN Convolutional Neural Network. 17

IDE Integrated Development Environment. 2

IoU Intersection over Union. 20

LSTM Long Short-Term Memory. 25

OCR Optical Character Recognition. 5

POD Page Object Detection. 26

R-CNN Fast Region-based Convolutional Neural Network. 17

REST REpresentational State Transfer. 2

SVM Support Vector Machine. 18

Bibliografia

Corsi frequentati

- [22] Andrew Ng. *Machine Learning*. URL: <https://www.coursera.org/learn/machine-learning> (visitato il 15/09/2018) (cit. a p. 9).
- [24] Andrew Ng; Younes Bensouda Mourri; Kian Katanforoosh. *Deep Learning Specialization*. URL: <https://www.coursera.org/specializations/deep-learning> (visitato il 15/09/2018) (cit. a p. 9).
- [25] Andrew Ng; Younes Bensouda Mourri; Kian Katanforoosh. *Neural Networks and Deep Learning*. URL: <https://www.coursera.org/learn/neural-networks-deep-learning> (visitato il 15/09/2018) (cit. a p. 9).
- [26] Andrew Ng; Younes Bensouda Mourri; Kian Katanforoosh. *Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization*. URL: <https://www.coursera.org/learn/deep-neural-network> (visitato il 15/09/2018) (cit. a p. 9).
- [27] Andrew Ng; Younes Bensouda Mourri; Kian Katanforoosh. *Structuring Machine Learning Projects*. URL: <https://www.coursera.org/learn/machine-learning-projects> (visitato il 15/09/2018) (cit. a p. 9).
- [28] Andrew Ng; Younes Bensouda Mourri; Kian Katanforoosh. *Convolutional Neural Networks*. URL: <https://www.coursera.org/learn/convolutional-neural-networks> (visitato il 15/09/2018) (cit. a p. 9).

Persone citate

- [23] Andrew Ng. URL: <https://www.coursera.org/instructor/andrewng> (visitato il 15/09/2018) (cit. a p. 9).

Siti web consultati

- [1] *Django: The Web framework for perfectionists with deadlines*. URL: <https://www.djangoproject.com/> (visitato il 18/09/2018) (cit. a p. 1).
- [6] *The world's leading software development platform: GitHub*. URL: <https://github.com/> (visitato il 15/09/2018) (cit. alle pp. 2, 23).

- [9] *RiskAPP! la startup per la valutazione del rischio.* URL: <http://www.economia.rai.it/articoli/riskapp-la-startup-per-la-valutazione-del-rischio/31338/default.aspx> (visitato il 18/09/2018) (cit. a p. 4).
- [10] *Deloitte Digital Disruptors - STARTUP PROGRAMS.* URL: <http://www.deloittedigitaldisruptors.com/category/deloitte-digital-disruptors/> (visitato il 18/09/2018) (cit. a p. 4).
- [11] *MundiLab.* URL: <http://mundi-lab.com/> (visitato il 18/09/2018) (cit. a p. 4).
- [12] *FinTech Innovation Lab.* URL: <http://www fintechinnovationlab com/> (visitato il 18/09/2018) (cit. a p. 4).
- [13] *deloitte.* URL: <https://www2.deloitte.com/it/it.html> (visitato il 18/09/2018) (cit. a p. 4).
- [14] *Accenture.* URL: <https://www.accenture.com/it-it/new-applied-now> (visitato il 18/09/2018) (cit. a p. 4).
- [15] *Munich Re.* URL: <https://www.munichre.com/en/homepage/index.html> (visitato il 18/09/2018) (cit. a p. 4).
- [16] *Swiss Re.* URL: www.swissre.com/ (visitato il 18/09/2018) (cit. a p. 4).
- [17] *Fidelidade.* URL: <https://www.fidelidade.pt/> (visitato il 18/09/2018) (cit. a p. 4).
- [18] *P&V: Met u, altijd en overal.* URL: <https://www.pv.be/home> (visitato il 18/09/2018) (cit. a p. 4).
- [19] *STAGE-IT, intro aziende-studenti.* URL: <informatica.math.unipd.it/laurea/stageit.html> (visitato il 18/09/2018) (cit. a p. 5).
- [20] *Medium, where words matter.* URL: <https://medium.com/> (cit. a p. 9).
- [21] *Coursera.* URL: <https://www.coursera.org/> (cit. a p. 9).
- [29] Adam Geitgey. *Machine Learning is Fun!* Mag. 2014. URL: <https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471> (visitato il 13/09/2018) (cit. a p. 9).
- [30] Devin Soni. *Supervised vs. Unsupervised Learning.* Mar. 2018. URL: <https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d> (visitato il 13/09/2018) (cit. a p. 10).
- [35] Joyce Xu. *An Intuitive Guide to Deep Network Architectures.* Ago. 17. URL: <https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41> (visitato il 13/09/2018) (cit. alle pp. 13, 14).
- [36] Christian Szegedy; Wei Liu; Yangqing Jia; Pierre Sermanet; Scott Reed; Dragomir Anguelov; Dumitru Erhan; Vincent Vanhoucke; Andrew Rabinovich. *Going Deeper with Convolutions.* Set. 14. URL: <https://arxiv.org/abs/1409.4842> (visitato il 16/09/2018) (cit. a p. 14).
- [38] *COCO-trained models statistics.* URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md#coco-trained-models (visitato il 17/09/2018) (cit. alle pp. 21, 29).

- [52] Liangcai Gao; Xiaohan Yi; Leipeng Hao; Zhuoren Jiang; Zhi Tang. *ICDAR 2017 POD Competition*. Mar. 17. URL: http://www.icst.pku.edu.cn/cpdpcdar2017_PODCompetition/index.html (visitato il 16/09/2018) (cit. a p. 26).
- [53] Institute of Computer Science & Technology, Peking University. URL: <http://www.icst.pku.edu.cn/index.php?s=/Home/Index/index/lang/en.html> (visitato il 14/09/2018) (cit. a p. 26).
- [54] CiteCeerX è un motore di ricerca per paper scientifici e accademici nei campi della computer e information science. URL: <http://citeseerkx.ist.psu.edu/index> (visitato il 16/09/2018) (cit. a p. 26).
- [55] COCO is a large-scale object detection, segmentation, and captioning dataset. URL: <http://cocodataset.org/> (visitato il 16/09/2018) (cit. alle pp. 27, 29).
- [56] Open Images is a dataset of 9 million images that have been annotated with image-level labels and object bounding boxes. URL: <https://storage.googleapis.com/openimages/web/index.html> (visitato il 16/09/2018) (cit. a p. 27).
- [57] COCO is a large-scale object detection, segmentation, and captioning dataset. URL: <http://cocodataset.org/> (visitato il 16/09/2018) (cit. a p. 27).
- [58] AVA is a project that provides audiovisual annotations of video for improving our understanding of human activity. URL: <https://research.google.com/ava/> (visitato il 16/09/2018) (cit. a p. 27).
- [60] Distance transform example and meaning in OpenCV. URL: https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ga8a0b7fdfcb7a13dde018988ba3a43042 (visitato il 16/09/2018) (cit. a p. 27).
- [61] Using your own dataset. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/using_your_own_dataset.md (visitato il 16/09/2018) (cit. a p. 28).
- [62] Regularization for Simplicity: L2 Regularization. URL: <https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/l2-regularization> (visitato il 17/09/2018) (cit. a p. 29).
- [63] Anish Singh Walia. Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent. Giu. 2017. URL: <https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f> (visitato il 17/09/2018) (cit. a p. 29).
- [64] Rohith Gandhi. A Look at Gradient Descent and RMSprop Optimizers. Giu. 2018. URL: <https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b> (visitato il 17/09/2018) (cit. a p. 31).
- [65] Vitaly Bushaev. Stochastic Gradient Descent with momentum. Dic. 2017. URL: <https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d> (visitato il 17/09/2018) (cit. a p. 31).
- [66] Andrew Ng. Adam optimization algorithm. URL: <https://www.coursera.org/lecture/deep-neural-network/adam-optimization-algorithm-w9VCZ> (visitato il 17/09/2018) (cit. a p. 31).

- [67] *The Pipeline Pattern*. URL: <https://www.cise.ufl.edu/research/ParallelPatterns/PatternLanguage/AlgorithmStructure/Pipeline.htm> (visitato il 17/09/2018) (cit. a p. 33).
- [68] *Single Responsibility Principle*. URL: <https://www.oodesign.com/single-responsibility-principle.html> (visitato il 17/09/2018) (cit. a p. 33).
- [70] *Denoising Dirty Documents, Kaggle kernels*. URL: <https://www.kaggle.com/c/denoising-dirty-documents/kernels> (visitato il 17/09/2018) (cit. a p. 34).
- [71] *Kaggle is the place to do data science projects*. URL: <https://www.kaggle.com/> (visitato il 17/09/2018) (cit. a p. 34).

Prodotti software utilizzati

- [2] *React: A JavaScript library for building user interfaces*. URL: <https://reactjs.org/> (visitato il 18/09/2018) (cit. a p. 2).
- [3] *Redux is a predictable state container for JavaScript apps*. URL: <https://redux.js.org/> (visitato il 18/09/2018) (cit. a p. 2).
- [4] *Django REST framework*. URL: <http://www.django-rest-framework.org/> (visitato il 18/09/2018) (cit. a p. 2).
- [5] *PyCharm: Python IDE for Professional Developers*. URL: <https://www.jetbrains.com/pycharm/> (visitato il 15/09/2018) (cit. alle pp. 2, 23).
- [7] *IntelliJ: the most intelligent Java IDE*. URL: <https://www.jetbrains.com/idea/> (visitato il 18/09/2018) (cit. a p. 2).
- [8] *CircleCI: The shortest distance from idea to execution*. URL: <https://circleci.com/> (visitato il 18/09/2018) (cit. a p. 3).
- [31] *Keras: The Python Deep Learning library*. URL: <https://keras.io/> (visitato il 14/09/2018) (cit. alle pp. 12, 24).
- [32] *Tensorflow: An open source machine learning framework for everyone*. URL: <https://www.tensorflow.org/> (visitato il 14/09/2018) (cit. alle pp. 12, 17, 23, 24).
- [33] *Microsoft Cognitive Toolkit (CNTK), an open source deep-learning toolkit*. URL: <https://github.com/Microsoft/cntk> (visitato il 14/09/2018) (cit. a p. 12).
- [34] *Theano*. URL: <https://github.com/Theano/Theano> (visitato il 14/09/2018) (cit. a p. 12).
- [37] *Tensorflow object detection API*. URL: https://github.com/tensorflow/models/tree/master/research/object_detection (visitato il 14/09/2018) (cit. alle pp. 17, 24, 26).
- [39] *TableTrainNet: Table recognition inside documents using neural networks*. URL: <https://github.com/mawanda-jun/TableTrainNet> (visitato il 15/09/2018) (cit. a p. 23).
- [40] *IntelligentOCR: An intelligent OCR to detect tables and pure text inside PDFs and obtain a csv file and a txt from it*. URL: <https://github.com/mawanda-jun/IntelligentOCR> (visitato il 15/09/2018) (cit. a p. 23).

- [41] *Python*. URL: <https://www.python.org/> (visitato il 15/09/2018) (cit. a p. 23).
- [42] *Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors*. URL: <https://pillow.readthedocs.io/en/latest/> (visitato il 15/09/2018) (cit. a p. 24).
- [43] *OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library*. URL: <https://opencv.org/> (visitato il 15/09/2018) (cit. a p. 24).
- [44] *Alyn: Detect and fix skew in images containing text*. URL: <https://github.com/mawanda-jun/Alyn> (visitato il 15/09/2018) (cit. alle pp. 25, 34).
- [45] *Scikit-image is a collection of algorithms for image processing*. URL: <https://scikit-image.org/> (visitato il 15/09/2018) (cit. a p. 25).
- [46] *pdftoppm: Portable Document Format (PDF) to Portable Pixmap (PPM)*. URL: <https://www.systutorials.com/docs/linux/man/1-pdftoppm/> (visitato il 15/09/2018) (cit. alle pp. 25, 34).
- [47] *Use ImageMagick® to create, edit, compose, convert bitmap images*. URL: <https://www.imagemagick.org/> (visitato il 15/09/2018) (cit. alle pp. 25, 34).
- [48] *Tabula is a tool for liberating data tables locked inside PDF files*. URL: <https://tabula.technology/> (visitato il 15/09/2018) (cit. alle pp. 25, 34).
- [49] *pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language*. URL: <https://pandas.pydata.org/> (visitato il 15/09/2018) (cit. alle pp. 25, 28).
- [50] *Tesseract: an optical character recognition (OCR) engine*. URL: <https://opensource.google.com/projects/tesseract> (visitato il 15/09/2018) (cit. alle pp. 25, 34).
- [51] *Tensorflow Model Zoo*. URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md#coco-trained-models (visitato il 14/09/2018) (cit. a p. 26).
- [69] *Wand is a ctypes-based simple ImageMagick binding for Python*. URL: <http://docs.wand-py.org/en/0.4.4/> (visitato il 17/09/2018) (cit. a p. 34).

Articoli scientifici

- [59] Azka Gilani et al. «Table Detection Using Deep Learning». In: (set. 2017). URL: https://www.researchgate.net/publication/320243569_Table_Detection_Using_Deep_Learning (visitato il 16/09/2018) (cit. a p. 27).