UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Big Data Analytics and Business Intelligence

Una fondamentale applicazione di IA per la SmartCity: il riconoscimento di targhe

Anno Accademico 2020/2021

Relatore
Ch.mo prof. Vincenzo Moscato
Correlatore
Ing. Angelo Falvo

Candidato
Francesco Ottata
matr. M63001082

A Nonno Vincenzo

Introduzione

Il prototipo oggetto di questo elaborato è stato sviluppato nel corso dei cinque mesi svolti come tirocinante presso l' azienda *Netgroup SRL*, sotto la supervisione dell' Ing. Angelo Falvo il quale si è prestato come tutor aziendale, e del professor Vincenzo Moscato. La ricerca effettuata rientra nel più grande progetto di sviluppo di idee innovative volte alla trasformazione delle nostre città in Smart City grazie all' integrazione di sistemi di analisi di Big Data, quali ad esempio flussi video, integrati a sistemi IOT, di cui l' azienda si fa promotrice attivamente. Questa tematica, oltre che estremamente attuale, è fortemente stimolante in quanto è legata indossolubilemente ad uno sguardo verso il futuro, grazie al progresso esponenziale dell' intelligenza artificiale che sta rendendo possibile cose che, solo fino a poco tempo fa, sembravano quasi fantascienza. Il seguente lavoro è strutturato come segue. Dopo una breve introduzione nel primo capitolo ai concetti di smart-city,smart-mobility e relativamente a questt'ultimi di Big Data Analytics e IOT, approfondiremo il caso d ' uso del riconoscimento di targa che rientra nel più ampio ventaglio delle applicazioni

di intelligenza artificiale nel campo della mobilità. Introdurremo poi i sistemi ALRP, dandone una definizione e illustrando le fasi del processo che essi implementano, e confronteremo alcuni risultati sperimentali ottenuti in passato riguardo. Nel terzo capitolo verrà descritta nel dettaglio l' architettura prototipale realizzata in tutti i suoi componenti, specificando gli strumenti utilizzati e motivando le scelte progettuali fatte. Per ogni fase verranno riportarti schemi ed esempi di output per rendere chiari i passaggi effettuati. Infine nel quarto capitolo verranno riportati e commentati i risultati sperimentali ottenuti, specificando i casi d' uso che si è scelto di testare, analizzando i punti di forza e di debolezza del modello, e fornendo una breve panoramica dei possibili sviluppi futuri.

Indice

Capitolo 1

La mobilità nella grandi metropoli : verso una soluzione smart

1.1 L'intelligenza artificiale al servizio delle grandi città

Il termine smart city, ovvero città inteligente, è stato oggetto di una diffusione e di un'uso sempre maggiore nel corso dell' ultimo decennio, di pari passo con i concetti di digital tranformation e Internet of Things, abbrevviato IOT. L' IOT è quel fenomeno la cui nascita ha caratterizzato l' evoluzione digitale degli ultimi anni e che tutt' ora è al centro di quest' ultima, ed è strettamente legato ai cosidetti og-

getti intelligenti. Con questa terminologia non si fa più riferimento solo a computer, smartphone e tablet ma a tutti i dispositivi che ci circondano all'interno delle nostre case, al lavoro, nella vita di tutti i giorni,appunto nell' intera città. L'Internet of Things nasce proprio qui: dall'idea di portare nel mondo digitale gli oggetti della nostra esperienza quotidiana. Al centro di quest'evoluzione tecnologica vi è la sempre più rapida diffusione dei cosidetti Big Data, ovvero l'insieme di tutti i dati ricavati da un'estesa raccolta informativa e che richiedono di essere gestiti attraverso dei software analitici che fanno uso spesso dell' IA. Volendo entrare nello specifico, si definiscono Big Data quei dati che abbiano almeno una delle seguenticaratteristiche: **Volume**,**Veriditicà**,**Velocità**,**Varietà** e **Valore**.

La finalità di questo processo non è solo quella di rendere una citta smart nel solo senso di intelligente ma parliamo anche di un miglioramento sensibile di sostenibilità, organizzazione, qualità della vita e sicurezza per i cittadini. Volendo utilizzare la definizione fornita dall' Unione Europea, una città per essere smart deve includere 6 dimensioni:

- Smart People, ovvero la propensione a rendere i cittadini coinvolti e partecipi.
- Smart Governance, che si riflette in una sempre maggiore importanza del capitale umano e dei beni della comunità.

- Smart Economy, un'economia basata sulla partecipazione e sulla collaborazione e che punta su ricerca e innovazione
- Smart Living, nella misura di dare un' elevata importanza al benessere delle persone.
- Smart Environment, dando priorità ad uno sviluppo sostenibile con basso impatto ambientale ed efficienza energetica.
- Smart Mobility, ovvero adottare soluzioni di mobilità intelligente,al fine di diminuire i costi, rendere più sicura e controllata la circolazione

È in particolare su quest' ultimo aspetto che il presente lavoro si concentra,affrontando il problema sempre più sentito della circolazione urbana ed extraurbana nelle grande metropoli.

1.2 Smart City and Safe City : videosorveglianza avanzata

Il tema della mobilità si intreccia inevitabilmente con quello della sicurezza. Infatti è risaputo che il mezzo che presenta il più elevato tasso d' incidente è proprio l' auto, questo anche a causa della non curanza dei guidatori i quali non sempre rispettano le regole stradali e limiti di velocità in mancanza di controlli serrati. Essere una smart city significa dunque anche essere una safe city, un luogo in cui le persone si sentano

sicure sia per le strade pedonali che alla guida di auto e moto. A tal fine è necessario l' utilizzo di un sistema che prevenga gli eventi criminosi e che permetta sanzionarli tempestivamente. Ciò sarà possibile con l'implementazione di sistemi avanzati di videosorveglianza in grado di sfruttare la diffusione di IoT, 5G e Intelligenza Artificiale. Dunque, codiuvato da queste tecnologie, l' uso di telecamere di sorveglianza diventa una prerogativa fondamentale per la sicurezza urbana. Si tratta infatti di strumenti dalle molteplici applicazioni, tra cui monitoraggio del traffico, controllo dei limiti di velocità,identificazione di trasgressori, controllo del territorio e gestione remota di zone di parcheggio e di transito. Ovviamente a tutti questi vantaggi si accompagna un tema molto delicato e che non verrà approfondito in questa ricerca, ovvero quello della privacy. Infatti sebbene una sorveglianza capillare inibisce comportamenti pericolosi e criminosi, ma allo stesso tempo l'accesso all'enorme mole di dati raccolti deve essere regolato per non ledere la privacy delle persone. Tale tematica è quanto mai attuale in particolare nelle grandi città orientali che stanno entrando prepotentemente e sempre più massicciamente, nel corso ultimo ventennio, all' interno mondo dell' IOT.

1.3 Il riconoscimento automatico delle targhe e la sua importanza

Tra le molteplici applicazioni che il connubbio di IA e telecamere di sorveglianza ha, quella del riconoscimento di targhe per auto, moto, autobus e camion di vario tipo è sicuramente una delle più utili e versatili. Infatti dotare una telecamera del riconoscimento automatico della targa delle veicolo permette ad esempio di :

- regolare il controllo automatico degli accessi a determinate zone
- identificare e segnalare in tempo reale veicolo i cui guidatori hanno appena commesso un effrazione
- segnalare chi supera i limiti di velocità
- raccogliere informazioni riguardo ai veicoli presenti in una città, creando un archivio targhe che possa essere utile alle forze dell' ordine o alle compagnie assicurative.

Il flusso video e i relativi dati provenienti da molteplici telecamere inoltre rispettano tutte le caratteristiche precedentemente citate dei Big Data per cui è possibile fare di esse una fonte di infomazioni per l' analisi multimensionale e la Business Intelligence, ovvero quel processo di trasformazione di dati e informazioni in conoscenza. Con l' avanzare delle tecnologie IOT, dell' intelligenza artificiale e della diffusione del 5G, queste applicazioni raggiungeranno una sempre maggiore efficienza

ed efficacia per rendere la mobilità sempre più smart and safe. Il presente lavoro vuole dunque presentare una prima implementazione sperimentale di un sistema in grado di riconoscere ed estrapolare le targhe in foto o video tramite la potenza e la versatilità delle reti neurali e del deep learning, il cui scopo si è ipotezzato essere quello di monitorare zone speicifiche quali ingressi dei parcheggi o punti stradali ritenuti cardine del sistema viario.

Capitolo 2

I sistemi automatici di riconoscimento di targhe

2.1 Definizione di un sistema ALRP

Un sistema di analisi video in grado di leggere le targhe degli autoveicoli e integrare perfettamente i dati all' interno di processi operativi e di sicurezza del sito interessato e che funga eventualmente da sorgente dati per sistemi di business intelligence o di analisi multidimensionale è detto ALRP, ovvero Automatic License Plate Recognition. Le informazioni estratte possono essere utilizzate in molte applicazioni pratiche, come la riscossione automatizzata dei pedaggi, l'applicazione della legge sul traffico, il controllo degli accessi agli spazi privati e il monitoraggio del traffico stradale. Per tali motivi i sistemi ALPR è stato oggetto di ricerca frequentemente da quando si è cominciato ad

investire in applicazione di intelligenza artificiale. In condizioni reali, tuttavia, molte delle soluzioni esistenti non sono ancora totalmente affidabili in quanto funzionano bene solo in presenza di vincoli specifici. Un robusto sistema ALPR dovrebbe essere in grado di funzionare in scenari non vincolati tenendo conto di ambienti aperti, immagini di bassa qualità e variazioni di illuminazione. Infatti il grande limite di tale sistema è la dipendenza dalla qualità dell' immagine e sopratutto della posizione del veicolo rispetto alla telecamera. Nella pratica è molto probabile che se la targa non sia visibile in maniera frontale il sistema non sarà in grado di effettuare la detection e la classificazione di tutti i caratteri correttamente. Il primo passo per qualsiasi sistema ALPR consiste nell'individuare la regione o zona LP(License Plate) in un'immagine e quindi applicare su di essa una qualche forma di metodo di riconoscimento ottico dei caratteri (OCR) per ottenere il dati LP. Questo capitolo mira a presentare una spiegazione dettagliata di come funziona un tipico sistema ALPR e fornire un'analisi comparativa dei diversi processi utilizzati in alcune soluzioni sperimentali proposte nel passato e attualmente in uso.

2.2 Fasi del processo

Lo scopo di un ALPR è,come detto più volte, localizzare, riconoscere ed estrarre informazioni in immagini o sequenze di frame all ' interno di un video. Tale processo avviene generalmente tramite le seguenti

tre fasi di elaborazione:

- localizzazione della regione
- segmentazione dei caratteri di targa;
- riconoscimento di ciascun carattere

La combinazione degli ultimi due passaggi è indicata solitamente come OCR(optical characthers recognition). Si desidera un'elevata precisione per il primo passaggio, poiché il mancato rilevamento della zona LP porterà probabilmente a un errore nei passaggi successivi. Per ridurre al minimo i tempi di elaborazione e rimuovere i falsi positivi, diversi metodi cercano prima il veicolo e poi il suo LP. Quindi, è possibile generalizzare ancor di più il processo suddividendolo in tre sottoattività che formano una pipeline sequenziale ovvero **rilevamento del veicolo**, **rilevamento LP** e **OCR**.

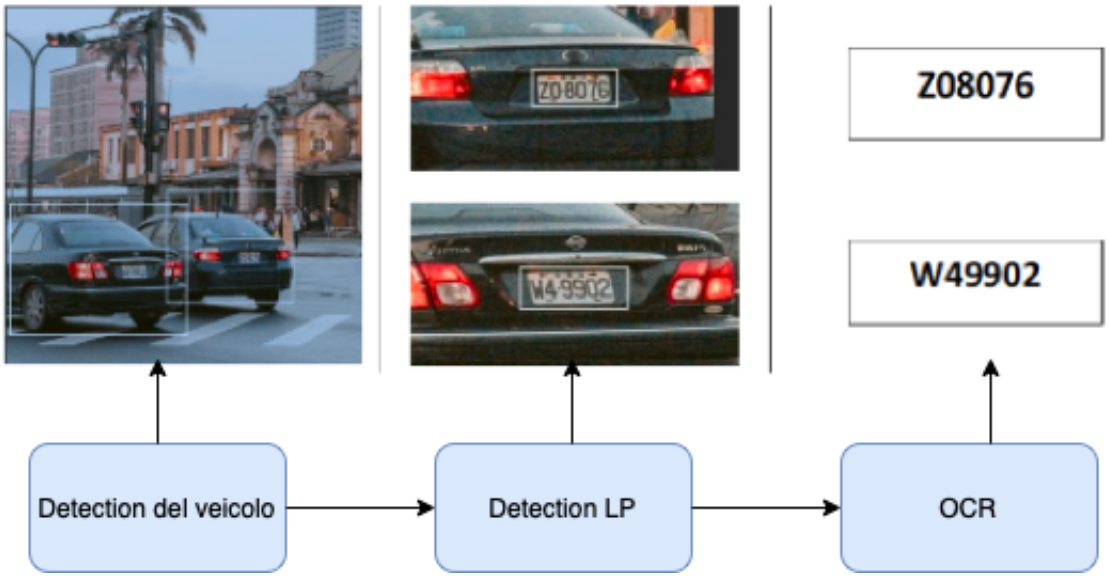


Figura 2.1: Pipeline del processo

Come anticipato il risulato di queste operazioni è influenzato da svariati fattori tecnici relativi al contenuto dell' immagine e alla sua qualità nonchè ad alcuni fattori ambientali. Una prima sfida consiste nel rendere capace il sistema di distinguere anche più targhe nella stessa immagine,cosi come di processare correttamente un' immagine in cui non vi è alcuna targa. La targhe potrebbero essere inoltre parzialmente oscurate da oggetti,o ancora non essere ben visibili a casua dell' angolazione della camera. Inoltre il modello deve essere addestrato a ricoscere targhe di varie forme,colori e dimensioni data la varietà delle stesse in base al paese di immatricolazione del veicolo. Grande importanza, come detto, hanno anche le varibili dovute all' ambiente in cui si trova la telecamera, in particolare le condizioni meteo e di illuminazione, ma anche la presenza di oggetti simili ad un targa nel background dell' immagine che potrebbero confondere il sistema.

2.2.1 Fase 1 : Detection del Veicolo

Il rilevamento dei veicoli è essenzialmente un compito di rilevamento degli oggetti, che a sua volta rientra nell 'ambito della computer vision e del rilevamento di istanze di oggetti semantici di una certa classe. È ampiamente riconosciuto che negli ultimi due decenni l'evoluzione del rilevamento degli oggetti ha attraversato due periodi storici: "periodo di rilevamento tradizionale degli oggetti (prima del 2014)" e "periodo di rilevamento basato sul deep learning (dopo il 2014)". La maggior

parte dei primi approcci al rilevamento degli oggetti si basava principalmente su caratteristiche artigianali individuate da un' esperto del dominio. Tuttavia, grazie alla disponibilità di set di dati annotati su larga scala e hardware (GPU) in grado di gestire una grande quantità di informazioni, la maggior parte dei moderni approcci di rilevamento degli oggetti implementa algoritmi di deep learning e sono stati in grado di ottenere un sostanziale aumento delle prestazioni, a tal punto che molti set di dati visivi su larga scala sono stati resi disponibili nel pubblico dominio. Molti di questi set di dati contengono numerosi oggetti di diverse categorie, mentre alcuni di loro si concentrano solo sui veicoli (o sulle automobili) in particolare. Nella figura 2.2 sono riportate dunque le principali tecniche utilizzate in entrambi gli approcci.

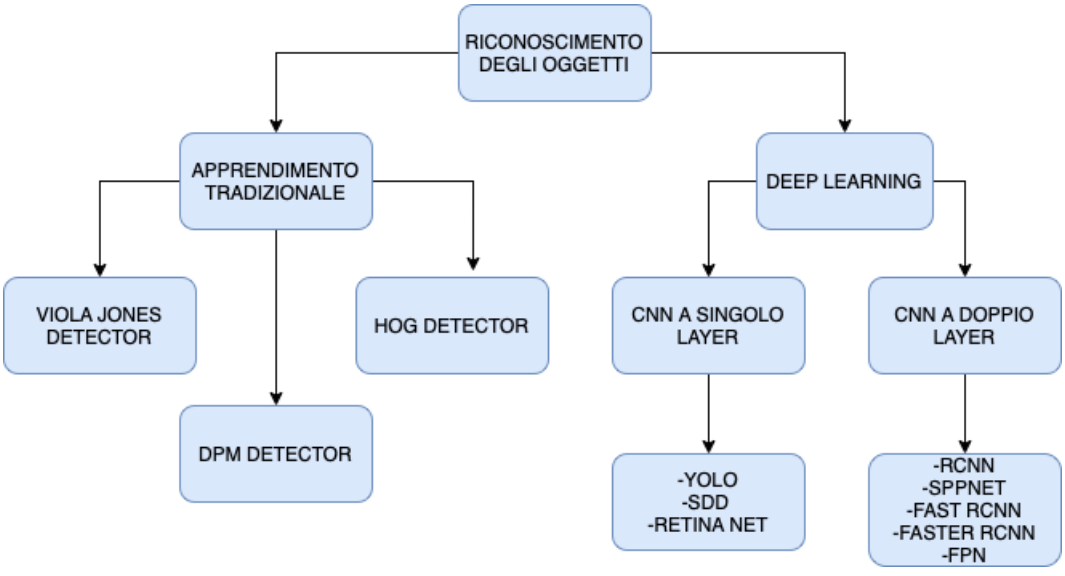


Figura 2.2: Principali tecniche di riconoscimento degli oggetti

2.2.2 Fase 2 : Detection LP

Il rilevamento della zona LP, simililmente al rilevamento dei veicoli, è anche essa un'attività di rilevamento degli oggetti. Pertanto, approcci similari a quelli della prima fase sono stati utilizzati per rilevare targhe all' interno delle immagini. In particolare molti algoritmi sono stati pensati per sfruttare il fatto che la quasi totalità delle targhe è di forma rettangolare. Dunque si fa uso di vari filtri e conversione dell' immagine ai fini del rilevamento dei bordi .

2.2.3 Fase 3 : OCR

La fase finale consiste nel segmentare la targa ed estrarre i caratteri tramite un algoritmo di OCR,come mostrato in figura 2.3.

Figura 2.3: Esempio del funzionamento di un OCR

2.2.4 Alcune proposte

Questa sezione vuole offrire una panoramica di alcuni sistemi ALPR [7] la cui documentazione è stata resa pubblica insieme ai metodi e alla pipeline di processo utilizzati in essi.

***"Reading Car License Plates Using Deep Convolutional Neural Networks"* di Li, Hui Shen, Chunhua** In questo documento del 2016 gli autori propongono una struttura a cascata in cui una CNN a 4 layer e in grado di riconoscere 37 classi (10 cifre, 26 lettere maiuscole e la categoria negativa non di caratteri) viene utilizzata nella prima fase per rilevare il testo in un'immagine. Quindi viene impiegato un altro classificatore CNN a 4 livelli per rifiutare i falsi positivi e distinguere i caratteri della targa dal testo generale. Una volta che quest' ultimi sono stati rilevati, la seconda fase utilizza una CNN a 9 layer per il riconoscimento dei caratteri, che è robusta a distorsioni, variazioni di illuminazione e rotazioni nell'immagine. Il loro metodo viene valutato su due set di dati, vale a dire il dataset Caltech Cars (Real) 1999[6] e il dataset di benchmark AOLP [9]. Per il set di dati Caltech Cars, questo metodo ha raggiunto una precisione dell'84,13% e del 92,07% rispettivamente per il rilevamento della targa e il riconoscimento dei caratteri. Per i sottoinsiemi AC, LE e RP di AOLP, è stata ottenuta la seguente precisione (in percentuale) rispettivamente per il rilevamento delle targhe e le attività di riconoscimento dei caratteri:

- AC (92,87, 98,38).
- LE (93,97, 98,19).
- RP (87,73, 96,56).

Tuttavia, anche con le prestazioni impressionanti nei risultati sperimentali, il metodo non è efficiente e non può essere utilizzato in scenari in tempo reale a causa della bassa velocità di rilevamento.

***"Deep Learning System for Automatic License Plate Detection and Recognition "* di Selmi, Zied Ben Halima, Mohamed Alimi** Nel 2017 un nuovo lavoro fu presentato dagli autori sopracitati.La relativa documentazione pubblicata sottolinea l'importanza delle fasi di pre-processing che vengono applicate all'immagine di input prima di passarla a un classificatore CNN targa/non targa. La pipeline di pre-processing utilizzata è la seguente:

- 1) Conversione RGB in immagine HSV.
- 2) Filtro morfologico per massimizzare il contrasto.
- 3) Filtro sfocatura gaussiana.
- 4) Soglia adattativa.
- 5) Rilevamento dei i contorni.
- 6) Filtro geometrico.

L'immagine risultante viene quindi passata alla CNN per il rilevamento della zona LP. L'output del rilevamento LP viene poi passato a un'altra pipeline di pre-processing per segmentare la targa ed infine

riconoscere tutti i caratteri utilizzando una seconda CNN con 37 classi. Sul dataset Caltech Cars , sono state ottenute una precisione del 93,8% e un'accuratezza del 94,8 % rispettivamente per il rilevamento LP e le attività OCR. Per i sottoinsiemi di AOLP invece la precisione (in percentuale) per il rilevamento LP e l'accuratezza (in percentuale) per l'OCR risulta essere:

- AC (92,6, 96,2).
- LE (93,5, 95,4).
- RP (92,9, 95,1).

***"Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks"* di H. Li, P. Wang and C. Shen** In questo articolo, gli autori hanno presentato una singola rete neurale unificata che esegue contemporaneamente sia il rilevamento LP che le attività di riconoscimento dei caratteri, utilizzando dunque un approccio diverso dai precenti lavori. I risultati ottenuti mostrano che incorporando il riconoscimento dei caratteri direttamente nella pipeline di rilevamento LP, il sistema risultante diventa più efficiente in termine di prestazioni ma meno in termini di accuratezza. Il vantaggio è che tale soluzione evita l'accumulo di errori intermedi se le attività di cui sopra sono state eseguite separatamente. I risultati sono stati valutati sui sottoinsiemi di AOLP ottenendo la seguente accuracy

(in percentuale) la quale risulta decisamente minore,come detto, per quanto riguarda lo step di riconoscimento dei caratteri.

- AC (95,29).
- LE (96,57).
- RP (83,63)

***"License Plate Detection and Recognition in Unconstrained Scenarios "* di Silva S.M., Jung C.R.** In questo articolo, gli autori hanno presentato un sistema ALPR basato sull' approccio deep learning che si concentra su scenari non vincolati, in cui la zona LP potrebbe essere notevolmente distorto a causa di viste oblique. Hanno introdotto una nuova rete neurale convoluzionale (CNN), denominata Warped Planar Object Detection Network (WPOD-NET[12]),la quale è stata utilizzata anche nel presente lavoro e per tanto ne verrà fornita una documentazione più dettagliata nel seguente capitolo.Tale rete è in grado di rilevare e annullare la deformazione di più targhe distorte in una singola immagine generando una matrice di trasformazione affine per ogni cella di rilevamento.Queste immagini rettificate vengono quindi inviate a un metodo di riconoscimento ottico dei caratteri (OCR) per ottenere il risultato finale. Il loro approccio ha raggiunto una precisione media dell'86,43% quando testato sul dataset OpenALPR (EU e BR) (disponibile su https://github.com/openalpr/openalpr), set di test SSIG, AOLP (RP) e il loro set di dati CD-HARD personalizzato

Approccio	Accuracy in percentuale	Accuracy in percentuale
CNN per entrambi i task	OpenALRP	LP Detection [86.89], Classificazione dei Caratteri[78.36]
Combinazione di algortimi per il rilevamento del contorno e Catene di Markov per l' OCR	Dataset custom	98.76
Combinazione di Yolo e CR-NET	SSIG	LP Detection [93.53], Classificazione dei Caratteri[78.33]
Combinazione di Yolo e rete GAN	AOLP	96.74

con solo dati aumentati a partire da quelli in lettura. La precisione media è aumentata all'89,33% per gli stessi set di dati quando hanno combinato dati reali aumentati con dati generati artificialmente per l'addestramento della rete OCR.

Confronto tra i risultati ottenuti Per completezza si riporta una tabella riassuntiva che mette a confronto alcuni metodi utilizzati e relativi risultati ottenuti.

Capitolo 3

La soluzione proposta: un approccio deep learning

3.1 Introduzione

In questo capitolo verrà illustrato l' approccio utilizzato,motivando le scelte fatte e fornendo una descrizione accurata di tutti gli strumenti utilizzati per implementare le fasi caratteristiche di un sistema ALRP. A valle degli studi fatti riguardo lo stato dell' arte del problema, la scelta è ricaduta sull' approccio che fa uso di deep learning e nello specifico della separazione delle fasi tramite l' utilizzo di più reti neurali.

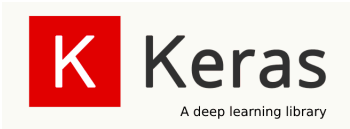
3.2 Linguaggio di programmazione,librerie, e ambiente di sviluppo

L' intero sistema è stato implementato in Python. Quest' ultimo infatti è diventato ormai il linguaggio di programmazione principe per lo sviluppo di applicazioni di intelligenza artficiale, machine learning, deep leaning e computer vision, grazie alla sua potenza e versatilità e all' integrazione con framework, librerie e strutture dati che mettono a disposizione innumerevoli strumenti per lavorare in questi campi. Nello specifico forniamo una descrizione dei più importanti tra questi e che per tanto sono alla base anche dell' intera architettura del sistema.

Scikit-learn [5] Scikit-learn (ex scikits.learn) è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento), support vector machine, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy. Scikit-learn si integra bene con molte altre librerie Python, come Matplotlib e plotly per la stampa, NumPy per la vettorizzazione degli array, i dataframe Pandas, SciPy e molte altre.

Keras [3]

Keras è una libreria scritta in Python (compatibile con Python2.7-3.6 e rilasciata sotto licenza MIT). Si tratta di software open source per l'apprendimento automatico e le reti neurali e supporta come back-end fra l'altro TensorFlow (dal 2017). Keras offre moduli utili per organizzare differenti livelli. Il tipo principale di modello è quello sequenziale, ovvero una pila lineare di layer. Keras permette una prototipazione facile e veloce, supporta sia reti convoluzionali (CNN) che reti ricorrenti (RNN) o combinazioni di entrambi, supporta schemi di connettività come multi-input e multi-output e funziona sia su CPU che GPU. Keras quindi è una libreria di alto livello, mentre Tensorflow (come Theano o CNTK) è una libreria di basso livello denominata backend, proprio per il ruolo che svolge nello stack software.



OpenCV [4] OpenCV nasce allo scopo di avere una base comune di strumenti analitici per immagini e video, primo dei quali una libreria che raccolga le funzionalità degli algoritmi più utilizzati, oltre che una serie di formati di rappresentazione dei dati secondo standard aperti e condivisi nel campo tanto affascinante quanto complesso della computer vision. L'utilizzo primario è infatti quello collegato alla visione artificiale, il cui problema principale, è quello di estrarre dalle immagini dati significativi e trattabili in modo automatico. La

libreria include attualmente più di 300 funzioni, che coprono le più svariate esigenze di trattamento di immagini, comprese funzioni matematiche ottimizzate (elevamento a potenza, logaritmi, conversioni cartesiane-polari, ecc.) ed un completo pacchetto di algebra matriciale, sviluppato funzionalmente al resto del sistema.

Google Colaboratory [1] Descriviamo infine l' ambiente di sviluppo utilizzato. Google Colaboratory, abbrevviato in Colab, è uno strumento gratuito presente nella suite Google che consente di scrivere codice python direttamente dal proprio browser. Si tratta di una piattaforma online che offre un servizio di cloud hosting per notebook Jupyter dove creare ricchi documenti che contengono righe di codice, grafici, testi, link e molto altro. Le macchine virtuali messe a disposizione in Google Colab ospitano un ambiente configurato che consente di concentrarsi sin da subito sui progetti di Data Science: sono presenti numerose librerie Python, tra cui moltissime e appena citate di Data Science come Keras e Tensorflow. Si può usufruire inoltre di GPU e TPU per dare boost computazionali importanti ai nostri lavori, per esempio nell'implementazione di reti neurali con Tensorflow. Tale caratteristica risulta molto comoda per eseguire task pesanti come l' elaborazione di un flusso video. Si sottolinea che in Google Colab le

risorse che vengono messe a disposizione agli utenti in maniera gratuita sono limitate, e variano a seconda delle fluttuazioni nella domanda. Per esempio un giorno si potrebbe venire assegnati ad una macchina virtuale con 32 GB di RAM e il giorno successivo ad una macchina virtuale con 12 GB di RAM. Se si ha bisogno di maggior stabilità, di macchine più performanti o di accedere a GPU e TPU più potenti, è possibile utilizzare la licenza a pagamento, ad oggi disponibile solamente negli Stati Uniti, presente la versione Pro di Google Colab.

3.3 Panoramica dell'architettura del sistema

Prima di entrare nel dettaglio delle varie fasi si vuole fornire uno sguardo generale sul sistema. Come detto in introduzione l' architettura è composta da due reti separate le quali svolgono rispettivamente la fase di detection di veicolo e zona LP e la fase di OCR che genera la stringa finale con i caratteri della targa. La prima rete è la già citata Wpod-net, il cui modello già addestrato è stato messa a dispozione dagli autori in maniera open-source ed è reperibile in svariate repository github. L' output di tale rete necessita di essere processato con l' applicazione di svariati filtri per essere poi elaborata da un metodo

che identifichi i contorni delle lettere. Solo a quel punto la zona LP con i contorni delle lettere oppurtunamente evidenziati viene data in ingresso ad una seconda rete, la quale è stata addestrata da zero su un dataset sufficientemente ampio contenente tutte le lettere dell' alfabeto e le 10 cifre decimali, che classificherà carattere per carattere e restituirà in uscita la stringa della targa. Si riporta dunque nella figura seguente una schematizzazione del tutto.

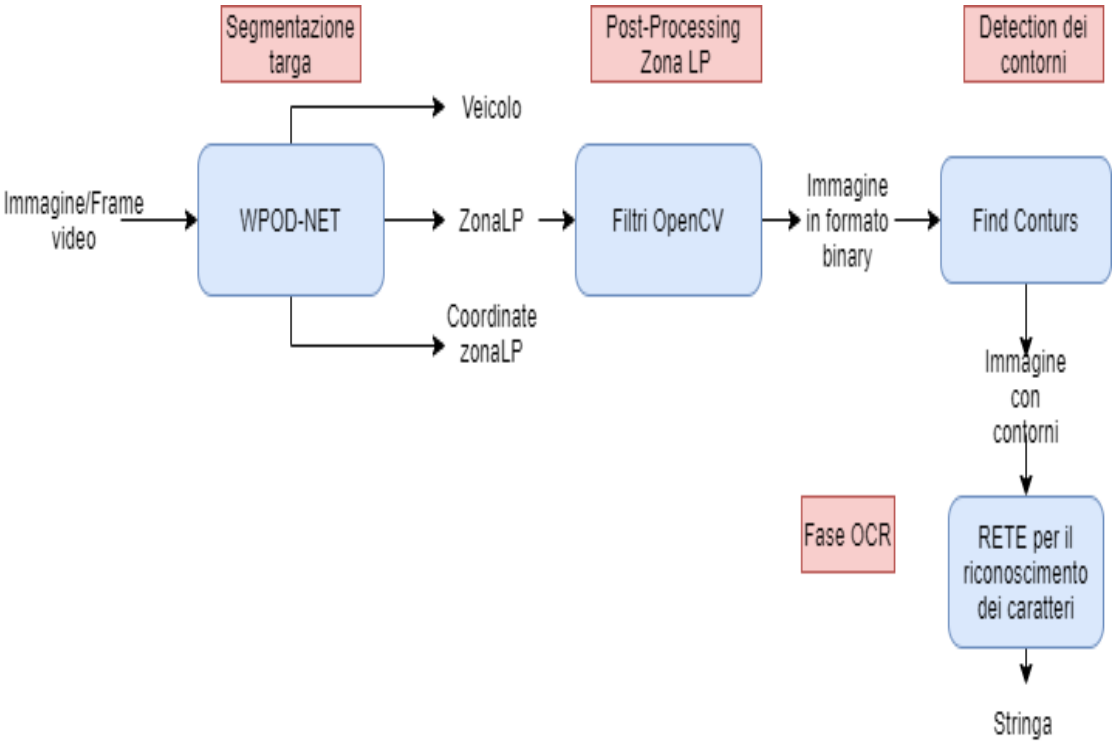


Figura 3.1: Schematizzazione del sistema

3.4 Wpod-Net

Forniamo adesso un approfondita descrizione di una delle due reti cardine del nostro sistema ALRP. WpodNet [12], acronimo di Warped Planar Object Detection Network è la rete che cerca la zona LP e la regredisce con una trasformazione affine per rilevamento, consentendo una rettifica dell'area in un rettangolo simile a una vista frontale. Questi rilevamenti possono esserre poi inviati inviati a una rete OCR per il riconoscimento finale dei caratteri. Poiché i veicoli sono uno delle classi di oggetti presenti in molti dataset di rilevamento e riconoscimento classici, come PASCAL-VOC, ImageNet e COCO, Wpod-Net,come specificato dagli autori nella documentazione della stessa, non è stata addestrata da zero ma a partire da un modello noto per eseguire il rilevamento del veicolo considerando alcuni criteri. Da un lato, si desiderava un alto tasso di richiamo, dal momento che qualsiasi veicolo non rilevato avente un LP visibile porta direttamente a un rilevamento errato complessivo. D'altra parte, è anche auspicabile un'elevata precisione per mantenere bassi i tempi di funzionamento, poiché ogni veicolo falsamente rilevato deve essere verificato da Wpod-Net. Sulla base di queste considerazioni, gli autori hanno deciso di utilizzare la rete YOLOv2[2] per la sua rapida esecuzione (circa 70 FPS) e la buona precisione e il compromesso di richiamo (76,8% sul set di dati PASCAL-VOC). Non è stata apportato alcuna modifica o perfezionamento a YOLOv2, ma è stata utilizzata la rete come una scatola

nera, unendo le uscite relative ai veicoli (ovvero auto e autobus), e ignorando le altre 79 classi che YOLO può identificare. I rilevamenti positivi vengono quindi ridimensionati prima di essere processati nella fase di detection della zona LP. Come regola generale, immagini di input più grandi consentono il rilevamento di oggetti più piccoli ma aumentano il costo computazionale. Nelle viste approssimativamente frontale/posteriore, il rapporto tra la dimensione LP e il riquadro di delimitazione del veicolo (BB) è elevato. Tuttavia questo rapporto tende ad essere molto più piccolo per le viste oblique/laterali, poiché il veicolo tende ad essere più grande e più allungato. Pertanto, le viste oblique dovrebbero essere ridimensionate a una dimensione maggiore rispetto a quelle frontali per mantenere ancora riconoscibile la regione LP. Sebbene metodi di stima possano essere usati per determinare la scala di ridimensionamento, questo lavoro presenta una procedura semplice e veloce basata sul rapporto di aspetto del veicolo. Quando è vicino ad 1, è possibile utilizzare una dimensione più piccola, che deve essere aumentata all'aumentare delle proporzioni. Più precisamente, il fattore di ridimensionamento *fsc* usato è dato da:

$$f_{sc} = \frac{1}{\min\{W_v, H_v\}} \min\left\{ D_{min} \frac{\max(W_v, H_v)}{\min(W_v, H_v)}, D_{max} \right\}$$

dove dove Wv e Hv sono rispettivamente la larghezza e l'altezza del veicolo. Si noti che $Dmin \leq$ fsc min(Wv,Hv) $\leq Dmax$, in modo che $Dmin$ e $Dmax$ delimitano l'intervallo per la dimensione più piccola

del ridimensionato. Sulla base di esperimenti e cercando di mantenere un buon compromesso tra precisione e tempi di esecuzione, sono state selezionate Dmin = 288 e Dmax = 608 di default, ma tali valori possono essere variati in caso di necessità. Dunque l'efficienza di questa rete è dovuta al fatto che impara a rilevare gli LP in una varietà di distorsioni diverse e regredisce i coefficienti di una trasformazione affine che "srotola" la zona LP distorto in una forma rettangolare simile a una vista frontale. Sebbene si possa utilizzare una proiezione prospettica planare invece della trasformata affine, la divisione coinvolta nella trasformazione prospettica potrebbe generare piccoli valori nel denominatore, e quindi portare a instabilità numeriche. Oltre a Yolo, le altre reti a partire dalle quali Wpod-Net è stata implementata sono SSD e STN. YOLO e SSD eseguono il rilevamento e il riconoscimento rapido di più oggetti contemporaneamente, ma non tengono conto delle trasformazioni spaziali, generando solo riquadri di delimitazione rettangolari per ogni rilevamento. Al contrario, STN può essere utilizzato per rilevare regioni non rettangolari, tuttavia non può gestire più trasformazioni contemporaneamente, eseguendo solo una singola trasformazione spaziale sull'intero input. Per chiarezza si riporta il processo di rilevamento, il quale è illustrato nella figura seguente, presente nella documentazione ufficiale.

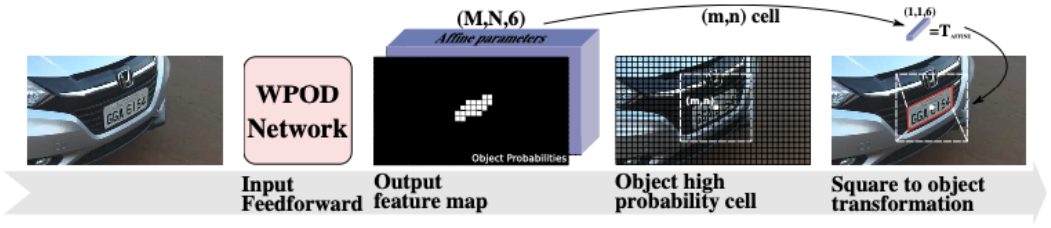


Figura 3.2: Processo di rilevamento di Wpod-Net

Inizialmente, la rete è alimentata dall'uscita ridimensionata del modulo di rilevamento del veicolo. Il feed forwarding si traduce in una mappa delle caratteristiche a 8 canali che codifica le probabilità oggetto/non oggetto e i parametri di trasformazione affine. Per estrarre la zona LP deformata, consideriamo prima un quadrato immaginario di dimensione fissa attorno al centro di una cella (m, n). Se la probabilità dell'oggetto per questa cella è al di sopra di una data soglia di rilevamento, parte dei parametri regrediti viene utilizzata per costruire una matrice affine che trasforma il quadrato fittizio in una regione LP. Pertanto, possiamo facilmente srotolare l'LP in un oggetto allineato orizzontalmente e verticalmente.

3.4.1 Architettura della rete

L'architettura proposta ha un totale di 21 strati convoluzionali, di cui 14 all'interno di blocchi residui. La dimensione di tutti i filtri convoluzionali è fissata in 3×3. Le attivazioni ReLU vengono utilizzate in tutta la rete, tranne nel blocco di rilevamento. Ci sono 4 livelli max

pooling di dimensione 2 × 2 e uno stride 2x2 che riduce la dimensionalità di input di un fattore 16. Infine, il blocco di rilevamento ha due livelli convoluzionali paralleli:

- Uno per inferire la probabilità, attivato da una funzione softmax.
- Uno per regredire i parametri affini, senza attivazione (o, equivalentemente, usando l'identità $F(x) = x$ come funzione di attivazione).

Si riporta una figura esplicativa nuovamente presa dalla documentazione ufficiale.

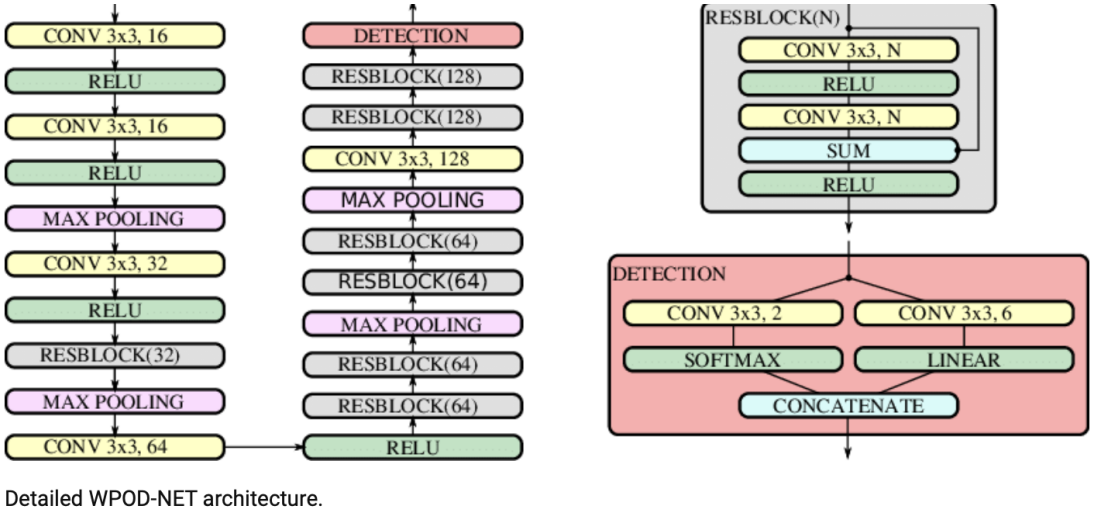


Figura 3.3: Architettura dettagliata di Wpod-Net

3.4.2 Funzione di Loss

Sia $p_i = [x_i, y_i]^T$, per i da 1 a 4, indichiamo i quattro vertici di una zona LP rilevata, in senso orario partendo dall'alto a sinistra. Inoltre, siano

$q_1 = [-0.5, -0.5]^T$,$q_2 = [0.5, -0.5]^T$, $q_3 = [0.5, 0.5]^T$, $q_4 = [-0.5, 0.5]^T$ i punti che denotano i corrispondenti vertici di un quadrato standard centrato nell'origine. Per un'immagine di input con altezza H e larghezza W e processata dalla rete data da 24 layer (quattro livelli massimi di pooling), la mappa delle funzionalità di output è costituita da un volume M × N × 8, dove M = H/N e N = W/N. Per ogni cella punto (m,n) nella mappa delle caratteristiche, ci sono otto valori da stimare: i primi due valori (v1 e v2) sono le probabilità oggetto/non oggetto e gli ultimi sei valori (v3 a v8) sono usati per costruire la trasformazione affine locale data da:

$$T_{mn}(\boldsymbol{q}) = \begin{bmatrix} \max(v_3, 0) & v_4 \\ v_5 & \max(v_6, 0) \end{bmatrix} \boldsymbol{q} + \begin{bmatrix} v_7 \\ v_8 \end{bmatrix},$$

dove la funzione max utilizzata per v3 e v6 è stata adottata per fare in modo che la diagonale sia positiva (evitando immagini speculari indesiderati o rotazioni eccessive). Per corrispondere alla risoluzione dell'uscita di rete, i punti $\mathbf{p_i}$ vengono ridimensionati e ricentrati in base a ciascun punto (m, n) nella mappa delle caratteristiche. Ciò si ottiene applicando una funzione di normalizzazione:

$$A_{mn}(\boldsymbol{p}) = \frac{1}{\alpha} \left(\frac{1}{N_s} \boldsymbol{p} - \begin{bmatrix} n \\ m \end{bmatrix} \right),$$

dove è una costante di scala che rappresenta il lato del quadrato immaginario. Il valore a cui è posto è 7,75, che è il punto medio tra le dimensioni LP massima e minima nei dati di allenamento aumentati

diviso per il learnig rate. Assumendo che ci sia un oggetto LP alla cella (m, n), la prima parte della funzione di Loss considera l'errore tra una versione deformata del quadrato canonico e i punti annotati normalizzati del LP, dato da:

$$f_{affine}(m,n) = \sum_{i=1}^{4} \|T_{mn}(\boldsymbol{q}_i) - A_{mn}(\boldsymbol{p}_i)\|_1.$$

La seconda parte della funzione di loss gestisce la probabilità di avere/non avere un oggetto nella cella (m,n). È simile alla perdita di confidenza della SSD, ed è fondamentalmente la somma di due funzioni di Loss logartimiche :

$$f_{probs}(m,n) = \text{logloss}(\mathbb{I}_{obj}, v_1) + \text{logloss}(1 - \mathbb{I}_{obj}, v_2),$$

dove I_obj è la funzione indicatore dell'oggetto che restituisce 1 se c'è un oggetto nel punto (m, n) o 0 in caso contrario. Un oggetto è considerato all'interno di una cella (m, n) se il suo rettangolo di delimitazione presenta un IoU maggiore di una soglia ${}_obj$ (impostata empiricamente a 0.3) che identifica un altro rettangolo di selezione della stessa dimensione e centrato in (m, n). La funzione di Loss finale è data da una combinazione dei termini definiti precedentemente ed è la seguente:

$$loss = \sum_{m=1}^{M} \sum_{n=1}^{N} [\mathbb{I}_{obj} f_{affine}(m,n) + f_{probs}(m,n)].$$

3.4.3 Addestramento della rete

Per completezza si riporta anche quello che è stato il processo di training di Wpod-Net. A questo scopo, gli autori hanno creato un set di dati con 196 immagini di cui 105 dal set di dati Cars[10], 40 dal set di dati SSIG[13] (sottoinsieme di addestramento) e 51 dal set di dati AOLP (sottoinsieme di LE). Per ogni immagine sono state annotati manualmente i 4 angoli della zona LP (a volte più di uno). Le immagini selezionate dal Cars Dataset includono principalmente targhe europee, ma ce ne sono molte dagli Stati Uniti. Le immagini di SSIG e AOLP contengono rispettivamente targhe brasiliane e taiwanesi. Dato il numero ridotto di immagini annotate nel set di dati di addestramento, è stato fatto largo uso della data augmentation. Nello specifico vengono utilizzate le seguenti trasformazioni:

- Rettifica: l'intera immagine viene rettificata in base all'annotazione LP, supponendo che la zona LP giaccia su un piano.
- Aspect Ratio: l' aspect ratio LP è impostato casualmente nell'intervallo [2,4] per adattarsi alle dimensioni di diverse regioni.
- Centratura: il centro della zona LP diventa il centro dell'immagine.
- Ridimensionamento: la zona LP viene ridimensionata in modo che la sua larghezza corrisponda a un valore compreso tra 40px e 208px (impostato sperimentalmente in base alla leggibilità de-

gli LP). Questa gamma viene utilizzato per definire il valore di utilizzato nell'equazione precedementente illustrata.

- Rotazione: viene eseguita una rotazione 3D con angoli scelti a caso, per contare per una vasta gamma di configurazioni della fotocamera.

- Mirroring: con 50% di possibilità;

- Traduzione: traduzione casuale per spostare il LP dal centro dell'immagine, limitato a un quadrato di 208 $\times$ 208 pixel attorno al centro.

- Ritaglio: considerando il centro LP prima della traduzione, viene ritagliata una regione intorno ad esso.

- Spazio colore: lievi modifiche nello spazio colore HSV.

- Annotazione: le posizioni dei quattro angoli della zona LP vengono regolate applicando le stesse trasformazioni spaziali utilizzate per aumentare l'immagine in ingresso.

Grazie all' utilizzo di queste trasformazioni sono state ricavate un gran varietà di immagini al fine di ottenere un training set completo. Si riporta, ancora una volta a partire dalla dettaglita documentazione ufficiale, un esempio dell' applicazione delle trasformazioni sopracitate.

Figura 3.4: Data Agumentation applicata ad un' immagine campione

La rete è stata dunque addestrata con 100.000 iterazioni di mini-batch di dimensione 32 utilizzando l'ottimizzatore ADAM. Il learnign rate è stato impostata su 0,001 con i parametri $1 = 0{,}9$ e $2 = 0{,}999$. I mini-batch sono stati generati scegliendo casualmente e aumentando i campioni dal training set, generando nuovi tensori di input di dimensione $32 \times 208 \times 208 \times 3$ ad ogni iterazione.

3.4.4 Esempio di output di Wpod-Net

A valle di questa prima fase riportiamo un' output di esempio fornito da WPOD-NET in fase di testing su una targa italiana. La targa è stata oscurata per motivi di privacy.

Figura 3.5: Output di Wpod-Net

Si sottolinea che l 'immagine in questione presenta una posizione ottimale della zona LP rispetto alla camera, per tanto il riconoscimento avviene perfettamente. Test più approfonditi richiedono di sottoporre alla rete immaggini di minore qualità,sebbene come spiegato precedentemente Wpod-Net riesce ad ottenere ottimi risultati anche nel caso di zone LP trasversali,oblique, e poco visibili.

3.5 Utilizzo di Open CV

Questa fase intermedia , ovvero l' utilizzo di Open CV per filtrare e trasformare l' immagine è uno degli step evolutivi rispetto al modello proposto dagli autori di WPOD-NET che si è scelto di implementare per migliorare l' accuracy da loro raggiunta del 86% circa e di cui si è discusso nel precedente capitolo. L' obbiettivo è quello di processare l 'immagine al fine di ottenere un formato utile alla detection dei bor-

di e successivamente alla classificazione dei caratteri. Verrà descritta dunque in questa sezione la pipeline di filtri che è stata utilizzata a valle di alcune sperimentazioni. Per ulteriori approndimenti sui filtri usati si rimanda alla documentazione uffciale di openCV.

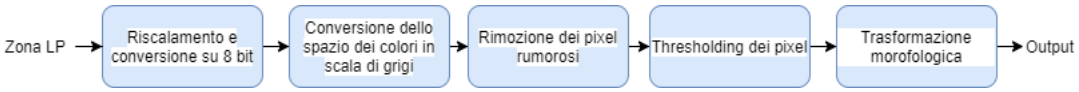


Figura 3.6: Pipeline Filtri Open CV

3.5.1 Riscalamento e conversione su 8 Bit

La funzione *cv::convertScaleAbs()* ridimensiona, calcola i valori assoluti e converte il risultato in 8 bit. La funzione è in realtà più funzioni riunite in una; essa esegue quattro operazioni in sequenza.

- La prima operazione consiste nel ridimensionare l'immagine sorgente in base ad un fattore alfa passato in ingresso come parametro.
- La seconda è compensare (aggiungere) di un fattore beta sempre passato come parametro.
- La terza è calcolare il valore assoluto.
- La quarta è convertire il risultato (con saturazione) in un carattere senza segno su 8 bit.

La grande efficienza di questa funzione è dovuta al fatto che quando si passano semplicemente i valori predefiniti (alpha = 1.0 o beta =

0.0), non è necessario avere timori per le prestazioni; OpenCV è abbastanza intelligente da riconoscere questi casi e non sprecare tempo del processore in operazioni inutili.

3.5.2 Conversione dello spazio dei colori in scala di grigi

Il metodo *cv2.cvtColor()* viene utilizzato per convertire un'immagine da uno spazio colore a un altro. Ci sono più di 150 metodi di conversione dello spazio colore disponibili in OpenCV. Nel nostro caso la conversione avviene in scala di grigi attraverso il passaggio del parametro cv2.COLORBGR2GRAY. Questa scelta è stata fatta al fine di eliminare problematiche dovute all' eventuale presenza di colori accesi le cui tonalità avrebbero potuto inficiare la successiva fase di detection dei contorni dei caratteri,i quali nel 90% dei casi sono di colore nero.

Riprendendo la targa di test mostrata in precedenza,il risultato a questo punto della pipeline viene mostrato nella seguente figura.

Figura 3.7: Conversione in scala di grigi

3.5.3 Rimozione dei pixel rumorosi

La sfocatura dell'immagine si ottiene convolundo l'immagine con un kernel filtro passa-basso. È utile per rimuovere il rumore. In realtà rimuove il contenuto ad alta frequenza (ad esempio: rumore, bordi) dall'immagine con conseguente sfocatura dei bordi quando viene applicato questo filtro. (Ci sono tecniche di sfocatura che non sfocano i bordi). OpenCV fornisce principalmente quattro tipi di tecniche di sfocatura:

- Averaging
- Gaussian Filtering
- Median Filtering
- Bilateral Filtering

Nel nostro caso è stata usata la seconda opzione,per cui ci limiteremo a fornire una descrizione della stessa.In questo approccio, invece di un filtro a scatola costituito da coefficienti di filtro uguali, viene utilizzato un kernel gaussiano.il quale è implementato con la funzione, *cv2.GaussianBlur()*. Si deve specificare la larghezza e l'altezza del kernel che dovrebbero essere positive e dispari. Dovremmo anche specificare la deviazione standard nelle direzioni X e Y, rispettivamente sigmaX e sigmaY. Se viene specificato solo sigmaX, sigmaY viene considerato uguale a sigmaX. Se entrambi vengono dati come zeri,

vengono calcolati dalla dimensione del kernel. Il filtraggio gaussiano è molto efficace nella rimozione del rumore gaussiano dall'immagine.

3.5.4 Thresholding dei pixel

Con questo filtro,implementato dal metodo *cv.threshold()*,per ogni pixel viene applicato uno stesso valore di soglia. Se il valore dei pixel è inferiore alla soglia, è impostato su 0, altrimenti è impostato su un valore massimo.*cv.threshold()* viene utilizzato appunto per applicare la soglia. Il primo argomento è l'immagine sorgente, che deve essere un'immagine in scala di grigi. Il secondo argomento è il valore di soglia utilizzato per classificare i valori dei pixel. Il terzo argomento è il valore massimo che viene assegnato ai valori di pixel che superano la soglia. OpenCV prevede diversi tipi di sogliatura che è data dal quarto parametro della funzione. La soglia di base descritta sopra viene eseguita utilizzando il tipo cv.THRESHBINARY. Tutti i tipi di soglia semplice sono:

- cv.THRESHBINARY
- cv.THRESHBINARYINV
- cv.THRESHTRUNC
- cv.THRESHTOZERO
- cv.THRESHTOZEROINV

Il metodo restituisce due output. La prima è la soglia utilizzata e la seconda uscita è l'immagine con soglia. Un' ulteriore tipologia di soglia utilizzabile è data dal metodo di Oshu.Nella soglia globale, abbiamo utilizzato un valore scelto arbitrariamente come soglia. Al contrario, il metodo di Otsu evita di dover scegliere un valore e lo determina automaticamente. Considerando un'immagine con solo due valori distinti (immagine bimodale), dove l'istogramma consisterebbe solo di due picchi. Una buona soglia sarebbe nel mezzo di questi due valori. Allo stesso modo, il metodo di Otsu determina un valore di soglia globale ottimale dall'istogramma dell'immagine. Per fare ciò, viene utilizzata semopre la funzione *cv.threshold()*, dove cv.THRESHOTSU viene passato come flag aggiuntivo. Il valore di soglia può essere scelto arbitrariamente. L'algoritmo trova quindi il valore di soglia ottimale che viene restituito come primo output,aumentando nettamente l' efficienza del filtro. Per tale motivo il nostro filtro è stato implementato facendo uso di tale soglia. Nella documentrazione ufficiale è riportata una sezione che illustra nel dettaglio come il metodo di Otsu determina la soglia. L' applicazione di questo filtro risulta fondamentale in quanto restituisce un' immagine binaria alla quale potremo poi applicare la detection dei contorni.

3.5.5 Trasformazione morfologica

L' ultimo step della pipeline che è stato applicato è una trasformazione morfologica,nello specifico la dilatazione,attraverso il metodo.Le trasformazioni morfologiche sono alcune semplici operazioni basate sulla forma dell'immagine. Normalmente vengono eseguite su immagini binarie. C'è bisogno di due input, uno è la nostra immagine originale, il secondo è chiamato elemento strutturante o kernel che decide la natura dell'operazione. Due operatori morfologici di base sono erosione e dilatazione. L'idea di base dell'erosione è proprio come l'erosione del suolo, erode i confini dell'oggetto in primo piano (cerca sempre di mantenere il primo piano in bianco). Quindi, cosa fa? Il kernel scorre attraverso l'immagine (come nella convoluzione 2D). Un pixel nell'immagine originale (sia 1 che 0) sarà considerato 1 solo se tutti i pixel sotto il kernel sono 1, altrimenti viene "eroso" (posto a zero). Quello che succede è che tutti i pixel vicino al confine verranno scartati a seconda della dimensione del kernel. Quindi lo spessore o la dimensione dell'oggetto in primo piano diminuisce o semplicemente diminuisce la regione bianca nell'immagine. È utile per rimuovere piccoli rumori bianchi o staccare due oggetti collegati. La dilatazione è esattamente l'opposto dell'erosione. Qui un elemento pixel è "1" se almeno un pixel sotto il kernel è "1". Quindi aumenta la regione bianca nell'immagine o aumenta la dimensione dell'oggetto in primo piano. Normalmente, in casi come la rimozione del rumore, l'erosione è seguita dalla dilatazio-

ne, perché l'erosione rimuove i rumori bianchi ma rimpicciolisce anche il nostro oggetto .Per tanto successivamente è necessario applicare la dilatazione. Avendo noi già utilizzato la rimozione del rumore tramite filtro gaussiano applichiamo solo la dilatazione. Come detto prima, il metodo *cv2.morphologyEx()*, richiede oltre al tipo di trasformazione morfologica che si vuole applicare anche il kernel per la convoluzione che funga da elemento strutturante. In OpenCV, possiamo utilizzare la funzione *cv2.getStructuringElement* stesso per definire il nostro elemento strutturante. A valle della pipeline di filtri dunque, l' output sarà il seguente.

Figura 3.8: Trasformazione morfologica

Grazie alle operazioni effettuate dunque, ottieniamo un input migliore da fornire ai successivi moduli del modello che eseguiranno la detection dei contorni e la classificazione dei caratteri.

3.6 Detection dei Contorni

Prima di passare alla fase di OCR per la classificazione dei caratteri, utilizziamo ancora le potenzialità di OpenCV. Infatti OpenCV ha molte funzionalità di elaborazione delle immagini che sono utili per

rilevare i bordi, rimuovere il rumore, applicare una soglia ecc. La funzione *FindContours()* recupera tutti i contorni nell'immagine che riesce a trovare. Ci possono essere vari modi in cui i contorni possono essere presenti nell'immagine. Alcuni ad esempio potrebbero essere annidati in altri contorni. Per facilitare la ricerca dei contorni che ci interessano e anche per conoscere la gerarchia in cui sono nidificati i contorni, i RetrievalModes sono molto importanti. L'output di RetrievalModes è un array Hierarchy che mostra come i vari contorni sono collegati tra loro, la loro relazione con altri contorni, relazione Parent Child. I parametri da fornire al metodo sono i seguenti:

- Image : un'immagine a canale singolo a 8 bit. I pixel diversi da zero vengono trattati come 1. I pixel zero rimangono 0, quindi l'immagine viene trattata come binaria qualora non lo fosse. È possibile utilizzare compare, inRange, threshold , AdaptiveThreshold, Canny e altri per creare un'immagine binaria da un'immagine in scala di grigi oa colori. Se la modalità è uguale a RETRCCOMP o RETRFLOODFILL, l'input può anche essere un'immagine intera a 32 bit di etichette (CV32SC1).

- Mode : Modalità di recupero del contorno.

- Method: Metodo di approssimazione del contorno.

Le scelte fatte per quanto riguarda modalità e metodo sono state cv2.RETREXTERNAL,che recupera solo i contorni esterni estremi,

e cv2.CHAINAPPROXSIMPLE che permette di memorizzare solo i punti d'angolo. Il metodo restituisce una lista contenete le coordinate dei contorni del singolo carattere e la gerarchia. Questo secondo parametro non rientra nel nostro interesse. Determinati i contorni il successivo step implementato è stato quello di utilizzare il metodo *cv2.boundingRect()* su ogni carattere salvato nella lista di contorni. Utilizzano dei vincoli in questo modo possiamo scartare tutti i contorni relativi a ciò che non sono i caratteri della targa, per quali possiamo settare ipoteticamente delle dimensioni standard,attraverso una parametro di ratio. Nello specifico sono state adottate le seguenti scelte. L' altezza standard del carattere è stata settata a 60, mentre la larghezza a 30. Il metodo *cv2.boundingRect()* restituisce altezza e larghezza di ogni riquadro tracciato attorno il relativo contorno. Determinato il ratio come altezza diviso larghezza ,vengono filtrate tutte le box che hanno un ratio compreso tra 1 e 3,5. Un ulteriore filtraggio viene fatto escludendo tutti i riquadri la cui altezza sia superiore alla metà dell' intera immagine da cui provengono. Solo a questo punto i caratteri i cui riquadri rispettano le condizioni imposte vengono separati a partire dall' immagine dilata ottenuta in precedenza e inseriti all' interno di una lista che verrà data in ingresso alla rete che eseguirà la classificazione. Tutti i passaggi descritti sono illustrati schematicamente in figura 3.9.

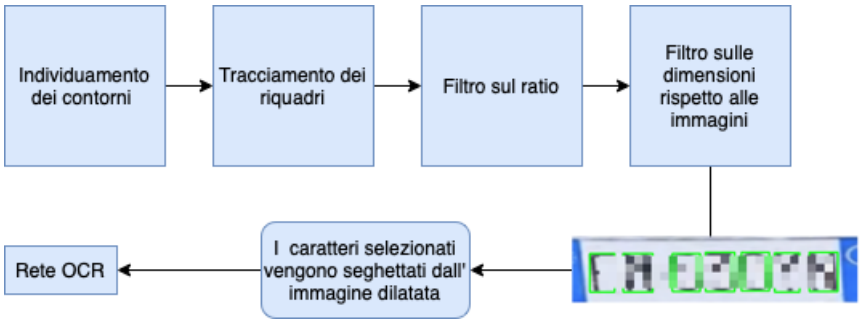


Figura 3.9: Schema delle fasi di selezione dei caratteri

3.7 Rete per il riconoscimento dei caratteri

Sulla base delle scelte di progetto effettuate, come illustrato in introduzione la seconda fase si un processo ALRP è stata implementata attraverso la realizzazione di una rete esterna a wpod-net,il cui addestramento è stato realizzato da zero. Illusteremo in questa sezione l' architettura della rete e i risultati ottenuti in fase di training, per poi illustrare come il modello è stato integrato nella pipeline generale.

3.7.1 Architettura della rete

Per realizzare la rete è stato fatto uso di keras,la libreria citata precedentemente. Il core della rete si compone di una rete parzialmente pre-configurata, MobileV2Net a cui sono stati aggiunti altri layer per adattare l' architettura al fine da noi voluto.

MobileNetV2 MobileNetV2[11] è un'architettura di rete neurale convoluzionale pensata per funzionare bene sui dispositivi mobili e che funge da feauture extractor. Si basa su una struttura detta residua invertita in cui le connessioni residue sono gli strati che sono dei colli di bottiglia. Lo strato di espansione intermedio utilizza leggere circonvoluzioni in profondità per filtrare le caratteristiche come fonte di non linearità. Nel complesso, l'architettura di MobileNetV2 contiene il livello iniziale di convoluzione completa con 32 filtri, seguito da 19 livelli residui. Una proprietà interessante di questa architettura è che fornisce una separazione naturale tra i domini di input/output dei blocchi residui (strati collo di bottiglia) e la trasformazione dei livelli, ovvero una funzione non lineare che converte l'input nell'output. La prima può essere vista come la capacità della rete ad ogni livello, mentre la seconda come l'espressività. Ciò è in contrasto con i tradizionali blocchi convoluzionali, sia regolari che separabili, dove sia l'espressività che la capacità sono intrecciate insieme e sono funzioni della profondità dello strato di output. Si riporta in figura 3.10 uno schema dei due tipi di layer che fungono il core del modello, come illustato nella documentazione ufficiale a cui si rimanda per approfondimenti.

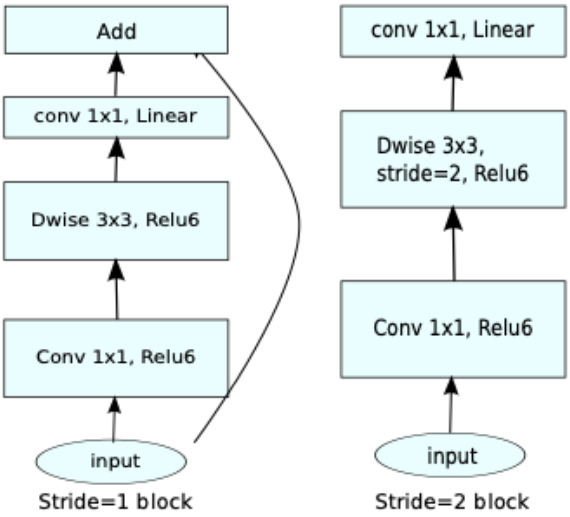


Figura 3.10: Layer di MobileV2Net

Layer aggiuntivi MobileV2Net,il cui modello è stato utilizzato a scatola chiusa,sebbene molto efficiente e prestante, ha richiesto alcune modifiche per essere ottimizata in quanto dovendo essere addestrata a riconoscere un gran numero di classi, ovvero tutti i caratteri dell' alfabeto inglese e le 10 cifre decimali per un totale di 36. Sono state ovviamente tenute in considerazione alcune informazioni, quale al esempio le dimensione delle immagine utilizzate per il trainig, le quali risultano di 28x28 su 3 canali. L' output desiderato è invece monodimensionale essendo un singolo carattere. A vale di un processo trial and error l' architettura definitiva ha previsto dunque l' aggiunta dei seguenti layer messi al base model.

- un layer *AveragePooling2D*,che sottocampiona l'input lungo le sue dimensioni spaziali (altezza e larghezza) prendendo il valore

medio su una finestra di input (di dimensione definita da poolsize) per ogni canale dell'input. La finestra viene spostata di passi lungo ogni dimensione.

- un layer *Flatten* che appiattisce l'input,ovvero rimodella il tensore per avere la forma che è uguale al numero di elementi contenuti nel tensore esclusa la dimensione batch.

- un layer *Dense* ovvero uno strato di rete neurale connesso in profondità, il che significa che ogni neurone nello strato dense riceve input da tutti i neuroni del suo strato precedente,nel nostro caso il flatten, e produce un' uscita della dimensione desiderata. Risulta essere lo strato più comunemente usato nei modelli. Matematicamente lo strato dense esegue una moltiplicazione matrice-vettore.

- un layer *Dropout*, che imposta casualmente le unità di input su 0,disattiavdole, con una frequenza settata al 50% ad ogni passaggio durante il training il che aiuta a prevenire il fenomeno dell' overfitting. Gli ingressi non impostati su 0 vengono aumentati di 1/(1 - rate) in modo che la somma di tutti gli ingressi sia invariata.

- Un' ultimo layer *Dense* che riduca l 'output ad una singola dimensione. A differenza del precedente non è stata utilizzata in questo caso come funzione di attivazione la classica ReLu ma

la Softmax. Un neurone con una funzione di attivazione ReLU accetta qualsiasi valore reale come suo input, ma si attiva solo quando questi input sono maggiori di 0. Di seguito è possibile trovare un grafico della funzione di attivazione ReLU.

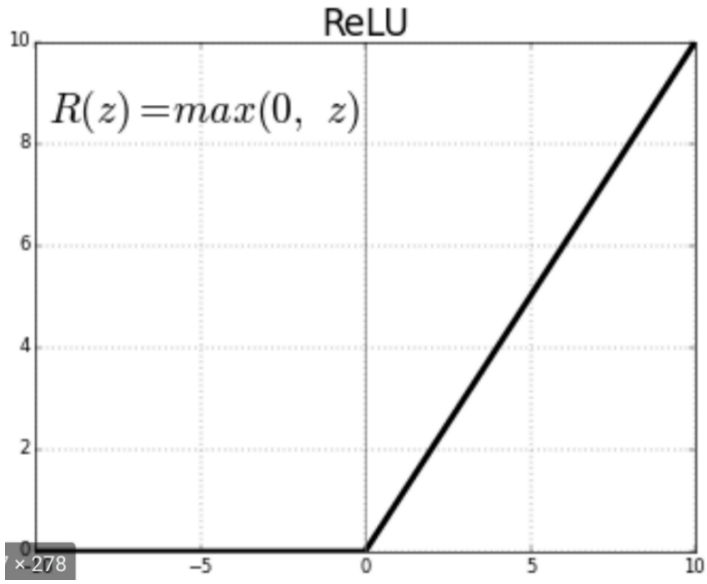


Figura 3.11: Funzione ReLu

La funzione Softmax Activation invece mappa gli input non normalizzati in un insieme di probabilità esponenziate e normalizzate. Nel contesto del Machine Learning, la funzione di attivazione Softmax viene utilizzata nei problemi di classificazione multi-classe per generalizzare la regressione logistica quando ci sono più di due classi di risultati,esattamente come nel nostro caso.

Dataset utilizzato Il modello è stata addestrato sul dataset **EMNIST** [8], un'estensione del più famoso **MNIST**. Tale ha lo scopo

di fornire più attività di riconoscimento ottico dei caratteri e quindi presenta i dataset do caratteri in cinque versione separate, denominate gerarchie di dati:

- ByPage: questa gerarchia contiene le scansioni binarie a pagina intera non elaborate dei moduli di esempio di scrittura a mano. I dati sui caratteri utilizzati nelle altre gerarchie sono stati raccolti attraverso un insieme standardizzato di moduli che gli scrittori sono stati invitati a compilare.In termine ultimo sono stati compilati 3699 moduli.

- ByAuthor: questa gerarchia contiene immagini di caratteri scritti a mano segmentati individualmente e organizzati per scrittore. Consente attività come l'identificazione dello scrittore, ma offre pochi vantaggi in termini di classificazione poiché ogni raggruppamento contiene cifre di più classi.

- ByField: questa gerarchia contiene le cifre e il carattere ordinati per campo nel modulo di raccolta in cui compaiono. Ciò è utile principalmente per segmentare le classi di cifre così come appaiono nei propri campi isolati.

- ByClass: rappresenta l'organizzazione più utile dal punto di vista della classificazione in quanto contiene le cifre ed i caratteri segmentati disposti per classe. Ci sono 62 classi che comprendo-

no [0-9], [a-z] e [A-Z]. I dati sono inoltre suddivisi in un insieme suggerito di formazione e test.

- ByMerge: questa gerarchia di dati risolve un problema interessante nella classificazione delle cifre scritte a mano, che è la somiglianza tra alcune lettere maiuscole e minuscole. In effetti, questi effetti sono spesso chiaramente visibili quando si esamina la matrice di confusione risultante dall'attività di classificazione completa sul set di dati ByClass. Questa variante sul set di dati unisce alcune classi, creando un'attività di classificazione di 47 classi. Le classi unite, come suggerito dal NIST, sono per le lettere C, I, J, K, L, M, O, P, S, U, V, W, X, Y e Z.

A causa dei limiti di potenza computazione e di spazio disponibile, è stata dunque da noi usata una versione semplificata del set ByClass, che contiene esclusivamente la distizione tra le 10 cifre e le 26 lettere maiuscole. Del resto questo non dovrebbe essere un problema essendo i caratteri delle targhe esclusivamente maiuscoli. Le dimensioni originali del dataset sono riportarte nella tabella seguente.

	Type	No. Classes	Training	Testing	Total
By Class	**Digits**	**10**	344,307	58,646	402,953
	Uppercase	**26**	208,363	11,941	220,304
	Lowercase	**26**	178,998	12,000	190,998

Tuttavia sia il training che il test set sono stati ulteriormente sottocampionati in maniera randomica ad un numero di immagini pari a

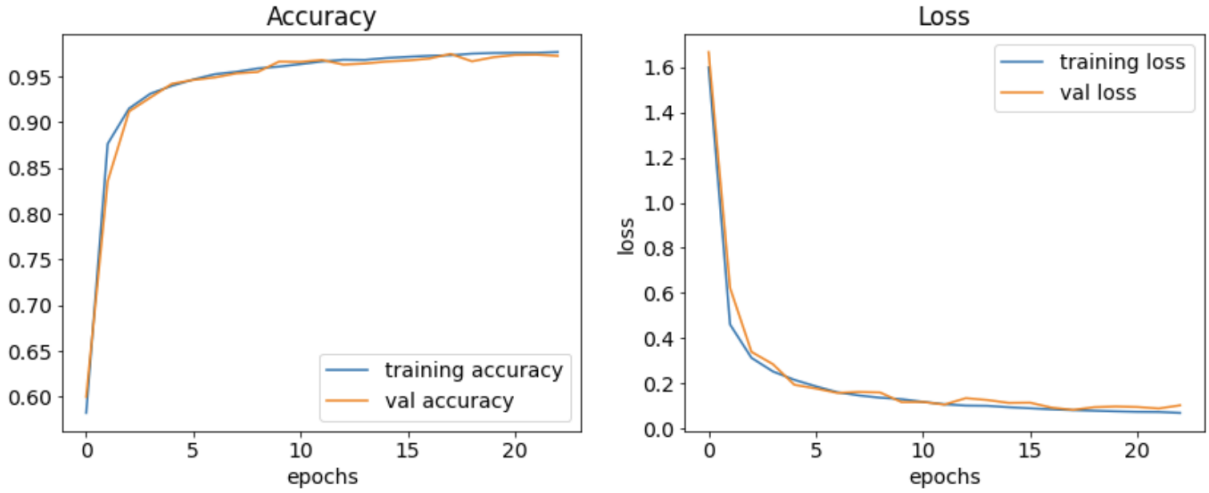
33387 e 10780 rispettivamente. A partire dal training set poi il 10% è stato estratto per la fase di validazione del modello, mantenendono una straficazione bilanciata delle classi.

3.7.2 Fase di addestramento

Essendo stato necessario sottocampionare pesantemente il dataset, in fase di training si è ricorsa alla tecnica di data augmentation, implementata grazie al metodo **DataGeneration()** messo a dispozione da Keras. Per la generazione di ulteriori immagini vengono effettuate rotazioni, traslazioni sia in altezza che larghezze, e zoom. Sono state effettuate diverse fasi di addestramento, effettuando manualmente il tuning degli iperparametri ad ogni tentativo in maniera trial error. Infatti a causa dei limiti tecnologici e della già elevata dispendiosità dell' addestramento, non sarebbe stato possibile implementare un tuning automatico tramite grid search. I parametri designati in definitiva sono stati dunque:

- Learning Rate : 1e-4
- Decay : Learnig Rate/Numero di Epoche
- Numero di epoche : 30
- Numero di batch : 64

Si riporta infine l'history dell' addestramento che mette in evidenza gli ottimi risultati ottenuti.



Il valore di accuracy ottenuto è stato dunque del 96% sul trainig set e del 92% sul test set. Il modello ed i relativi pesi sono stati dunque salvati ed esportati rispettivamente in formato json ed in formato h5 al fine di essere facilmente integrati nel sistema ALRP.

3.7.3 Classificazione dei caratteri

L' ultima fase del processo è dunque la classificazione dei singolo caratteri salvati nella lista precedentemente ottenuta. Una volta importato il modello, i pesi e codificate le label la classificazione avviene carattere per carattere. Nello specifico l'immagine contenente il carattere viene "stackata" in un vettore colonna attraverso funzioni della libreria numPy e viene aggiunta un terza dimensione attraverso la medesima libreria al fine di rendere compatibile l' immagine all' input richiesto dalla rete. La rete restituisce la classe predetta all' interno di una' array, per tanto il carattere viene convertito in stringa e concatenato ai pre-

cedenti. Una volta che l' intero array di caratteri è stato attraversato, la stringa rappresentante la targa viene stampata a schermo.

3.8 Analisi di flussi video

Un' evolutiva del prototipo sviluppato, necessaria all ' ipotetica applicazione reale del sistema su un impianto di telecamere che monitorino traffico o zone di parcheggio, è stata l' implementazione di un metodo che eseguisse tutto il processo descritto su tutti o su alcuni frame di un stream video,non in tempo reale ma registrato, dato in ingresso. Il numero di frame da analizzare infatti è un parametro variabile in base alla specifica applicazione. Ipotezzando che il video provenga da una strada a scorrimento veloce ad esempio, in cui le macchine viaggiono ad alta velocità, analizzare ogni singolo frame tenendo conto che non tutti possono avere la qualità necessaria ad una corretta predizione, è sicuramente la soluzione migliore. Se la telecamera fosse posta all' ingresso di un parcheggio invece analizzare tutti i frame sarebbe sicuramente rindonadante in quanto ve ne sarebbero diversi in cui sarebbe classificata la stessa targa. Dunque, è stato nuovamente utilizzato OpenCV per estrarre i singoli frame a partire da un flusso video e per ogni frame vengono eseguiti i passaggi precedentemente descritti. Ovviamente tutti i frame in cui non viene riconosciuta alcuna zona LP vengono scartati. Un problema da porsi è il seguente. Alcuni frame sicuramente risulteranno non adatti alla classificazione,in quanto ad

esempio la targa viene visualizzata per metà all' interno dell' immagine oppure in posizione eccessivamente obliqua per poter riconoscere tutti i caratteri. Pertanto come poter determinare automaticamente se un' output è corretto o sbagliato? Un prima soluzione implementata è stata quella di filtrare tutte le targhe classificate la cui lunghezza sia inferiore ai 7 caratteri. Questa tuttavia rende anche il sistema limitato alle solo targhe, ad esempio quelle italiane, che siano fatte di tale numero numero di caratteri. Ciò non è vero per altre targhe estere ad esempio. Si riporta in figura seguente uno schema esplicativo dell ' attuale processo di analisi dei flussi video.

Figura 3.12: Analisi dei flussi video

Capitolo 4

Test e risultati sperimentali

4.1 Creazione del dataset di testing

Al fine di testare il prototipo realizzato,è stata creato un piccolo dataset di videofile,utilizzando una piccola telecamera in grado di riprendere in risoluzione 4K, stabilizzata grazie ad un treppiedi. La telecamera è stata posta all' interno di un' area di parcheggio pubblica all' aperto, in due diverse posizioni al fine di testare diversi casi d' uso. Un primo test è stato fatto effettuando riprese da una distanza di circa 30 metri dalle auto, mentre un secondo test è stato effettuato da una distanza ravvicinata di 3 metri. Questo secondo caso d' uso è stato poi ulteriormente approfondito,sfruttando le diverse condizioni di luce e metereologiche nell' arco della giornata. La telecamera è stata posta

in maniera non totalmente frontale rispetto ai veicoli ma leggermente obliqua in maniera tale da rendere da testare l' efficienza del prototipo rispetto a tale problematica dei sistemi ALRP in generale, come spiegato nel capitolo 2. A causa dei già citati limiti tecnologici, ci si è limitati ad effettuare brevi riprese di massimo un minuto l' una, con soggetto un numero di auto che va da un minimo di 1 ad un massimo di 7. Si sottolinea che tutte si è ricevuto il consenso per ogni targa ripresa e che le riprese effettuate non verrano pubblicate in alcun modo per rispetto della privacy. Il dataset è dunque cosi composto:

- 10 minuti di video a 30 metri di distanza con meteo soleggiato in condizione di luce elevata.
- 10 minuti di video a 3 metri di distanza con meteo soleggiato in condizione di luce elevata.
- 10 minuti di video a 3 metri di distanza con meteo sereno in condizione di luce scarsa.
- 10 minuti di video a 3 metri di distanza con meteo nuvoloso.

4.2 Risultati ottenuti

Riportiamo dunque qui risultati ottenuti in termini di accuracy sia generali che suddivisi per casi d' uso. Ovviamente data la ristretta dimensione del dataset il valore di accuracy è stato possibile ricavarlo

etichettando manualmente le targhe. Inoltre si specifica nel calcolo dell' accuracy sono stati esclusi i rilevamenti non riusciti a causa di ostacoli eccessivamente grandi che impedivano una corretta detection della targa.

4.2.1 Risultati in relazione alla distanza

Distanza	Accuracy	Numero di veicoli
30 metri	83%	7
3 metri	95%	26
totale	89%	33

4.2.2 Risultati in relazione alle condizioni di luce

Condizioni di luce	Accuracy	Numero di veicoli
Luce elevata	100%	12
Luce scarsa	95%	7
Condizioni di pioggia	95%	7
totale	95%	26

4.2.3 Considerazioni a riguardo

I risultati ottenuti si possono definire incoraggianti, in quando sono similari e a tratti migliori rispetto ai lavori passati analizzati e a cui si è fatto riferimento. In particolare possiamo concludere che il post processing effettuato sull' output di WPOD NET migliora l' accuracy sopratutto nei casi più comuni in cui la targa è ben visibile e le condizioni meteo sono favorevoli. L' effecienza di wpod-net stessa inoltre rende il sistema performante anche nella casistica in cui la targa è in

posizione non perfettamente frontale rispetto alla camera e il filtraggio effettuato con OpenCV rende i caratteri quasi sempre distinguibili anche i condizione di luce meno ottimali. Notiamo che invece il sistema perde di precisione ed efficienza su distanza maggiori a 10 metri , avendo dunque difficoltà sia a segmentare la targa sia a distinguire correttamente tutti e 7 i caratteri.

4.3 Sviluppi futuri

Diverse migliorie sono apportabili al prototipo. In prima battuta effettuare dei test su dataset più ampi ed esplicativi di uno scenario reale potrebbe mettere in luce ancor di pù quelli che sono i punti di forza e quelli di debolezza dello stesso, in maniera tale da poter lavorare su quest' ultimi. Nello specifico una prima evolutiva da implementare potrebbe essere sicuramente un filtraggio più efficicace degli output corretti e quelli errati, che non si basi solamente sul numero di caratteri. Si potrebbe pensare di addestrare il modello non solo a riconoscere la targa ma anche la nazionalità della stessa. Dal punto di vista dell' esperienza dell ' utilizzatore è prevista invece lo sviluppo di un interfaccia grafica mobile o desktop in grado di interagire tramite API con il modello per effettuare la detection su video o immagine che vengono fornite in ingresso. Inoltre ipotizzando un ' uso reale del sistema, sicuramente si dovrebbe aumentare l' efficienza e le prestazioni al fine di poter analizzare flussi video inviati in streaming quasi in tempo reale, piuttosto che un solo video limitato per volta.

Conclusione

Al termine di questo lavoro si può concludere come gli strumenti odierni di IA, Machine Learning e Deep Learning insieme con le moderne tecnologie di trasmissione dati e da strumenti IOT sempre più perfomanti ed avanzati siano una base solida per la trasformazione delle nostre città nelle più moderne Smart City. Quello trattato è un campo le cui applicazioni sono potenzialmente infinite , dalla tutela della sizurezza nei luoghi pubblici al supporto ai sistemi di controllo per la mobilità urbana fino alla guida autonoma.La sfida resta quindi affinare gli algoritmi di riconoscimento immagini per poter essere utilizzati in un ventaglio di casi d'uso ben più ampio Dal punto di vista personale ho trovato la tematica molto interessante e stimolante,grazie anche al supporto dell' azienda nella figura dell' Ing. Falvo, ed è auspicabile che in un prossimo futuro il prototipo possa essere migliorato,completato e inserito in uno scenario reale di mercato nell' ambito di uno dei molti progetti innovativi che NetGroup porta avanti e di cui si fa promotrice,accompagnando le nostre città e i relativi cittadini in un futuro che in fondo non è cosi lontano.

Bibliografia

[1] *https://colab.research.google.com.*

[2] *https://github.com/dwaithe/yolov2.*

[3] *https://keras.io.*

[4] *https://opencv.org.*

[5] *https://scikit-learn.org/stable/".*

[6] Caltech plate dataset, 2003.

[7] Harshvardhan Bhatia, Aayush Narula, and Ms.Dhivya. A survey on automatic license plate recognition(alpr) systems. 2020.

[8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre´ van Schaik. Emnist: an extension of mnist to handwritten letters. *The MARCS Institute for Brain, Behaviour and Development*, 2017.

[9] G. Hsu, J. Chen, , and Y. Chung. Application-oriented license plate recognition, 2013.

[10] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. 2013.

[11] Mark Sandler, Andrew Howard, Menglong Zhu Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *Google Inc*, 2019.

[12] Sergio Silva and Claudio Rosito Jun. License plate detection and recognition in unconstrained scenarios. 2018.

[13] Gabriel Resende Gonc ̧alves, Sirlene Pio Gomes da Silvab, and David Menotti an d William Robson Schwartza. A benchmark for license plate character segmentation. 2016.

Ringraziamenti

Pochi ma sentiti ringraziamenti.

Grazie ai miei colleghi con cui ho iniziato e concluso questo difficile ma entusiasmante percorso: siamo cresciuti insieme, abbiamo superato mille difficoltà insieme, ci siamo divertiti e siamo sempre stati uno sprone l'uno per l' altro. Grazie in particolare a Daniel, più di un collega, un amico vero. Volevamo concludere insieme e ci è riuscito! Grazie, per ogni cosa, a Mamma, Papà e Martina: sapete quanto sono grato e orgoglioso di voi. Non c' è bisogno di aggiungere... Grazie ai miei fratelli di comunità, la mia seconda famiglia: da lontano mi avete osservato e accompagnato come solo un fratello sa fare. Grazie a chi, a modo suo, c'è stato sempre, dal primo esame ormai cinque anni fa fino all' ultimo giorno. Grazie a Nonno Vincenzo per cui non basterebbero mille parole e a Zio Marco: è stata dura concludere dopo che quest' anno mi avete lasciato; mi mancate molto ma so con certezza che siete sempre con me e da lassù state sorridendo osservandomi in questo giorno, orgogliosi. Grazie infine a me stesso. Mi piace essere umile ma devo ammettere che con molta forza di volontà, un po' di furbizia, un po' di fortuna e sopratutto tanti sacrifici, sono stato davvero bravo.

Spero che questa determinazione mi accompagni oggi e sempre.