UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

---

Sviluppo di un framework basato su algoritmi di Deep Learning per il monitoraggio di cantieri industriali

Development of a framework based on Deep Learning algorithms for monitoring industrial sites

Candidato:
Federico Colleluori

Relatore:
Prof. Adriano Mancini

Correlatore:
Dott. Riccardo Rosati
Dott. Flavio Tonetto

Anno Accademico 2021-2022

Università Politecnica delle Marche

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

Ringraziamenti

A tutti coloro che mi sono stati vicini in un modo o nell'altro e che mi hanno aiutato a migliorare.
A mia sorella che mi è stata a fianco.
Ai miei per il supportato, le attenzioni e l'affetto mostrati.
Ai nonni e gli zii che hanno incoraggiato le mie capacità.
Ai miei compagni di viaggio in questo duro percorso.
All'esperienza erasmus e agli amici di ESN, a riprova del fatto che "Once erasmus always erasmus".

Sommario

Il framework creato si occupa di automatizzare i compiti di monitoraggio e di sorveglianza delle condizioni di sicurezza all'interno di un'area cantieri oli & gas.
I droni sono lo strumento impiegato per la raccolta delle foto che, ad un certo istante nel tempo, catturano lo stato corrente dei lavori.
Tramite l'uso del software QGIS e di apposite classi python, queste immagini sono poi elaborate nella fase di pre-processamento. Essa mira a creare i profili del DSM, ovvero un modello digitale della superficie relativo al volo considerato.
Il dateset per il training dei modelli di reti neurali convoluzionali mono-dimensionali (1D CNN) è composto da alcuni dei voli già effettuati mediante droni, e che presentano una maggiore eterogeneità fra le classi.
L'obiettivo ultimo è quello di creare modelli che identificano tre classi differenti lungo tutta l'estensione del tracciato lavori: scavo aperto, scavo chiuso e posa del tubo. Pertanto, grazie a tecniche di data augmentation, personalizzazione di una classe, ottimizzazione degli iperparametri e splitting del dataset, si estende una pre-esistente classificazione binaria per crearne una multi-classe.
Nei diversi esperimenti si è voluto generalizzare i casi analizzati su un medesimo volo, su un diverso volo dello stesso cantiere, e infine sul volo di un altro cantiere.
I risultati hanno mostrato che nei modelli creati si raggiungono facilmente eccellenti livelli di accuratezza delle predizioni per il test su un sottoinsieme del dataset di training del modello, e con qualche marginale differenza, per il test su voli differenti nell'area del cantiere considerato. Gli esiti che riguardano l'ultima casistica si differenziano dalle due precedenti perché i modelli sono generati tramite dataset relativi ad un cantiere, e poi testati su dataset provenienti da un altro cantiere; qui si è sperimentato che un aumento consistente (il doppio o più) dei dati reali forniti al modello per il training, ad esempio ricavati dall'unione dei dataset di due voli, apportano un aumento dell'accuratezza generale sulle predizioni con un minor squilibrio tra le classi minoritarie.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Nel contesto di un'impresa di lavori su cantieri oli & gas si inserisce questo lavoro di sviluppo di un framework che automatizzi alcuni dei compiti del personale direzione lavori. A tal proposito, si intende rendere più efficiente e velocizzare il monitoraggio dell'evoluzione lavori nel tempo, oltre alla sorveglianza delle condizioni di sicurezza all'interno dell'area cantiere. Questi incarichi necessitano di numerosi e ripetitivi stadi, con annesso l'impiego di risorse umane, mezzi di trasposto e materiale tecnico.

Ci si domanda a questo punto se e come sia possibile automatizzare questi processi. Chiaramente nella realtà moderna e digitalizzata del XXI secolo l'orizzonte verso soluzioni di lavoro innovative e tecnologiche è molto ampio. Dunque, decisioni orientate all'utilizzo di nuove tecnologie, come i droni, è più che mai attuale. Questi velivoli comandati a distanza risultano particolarmente adatti al monitoraggio dello stato lavori in un cantiere perché in breve tempo sono capaci di percorrere grandi distanze e di catturare numerose immagini con diverse prospettive.
Partendo da queste immagini, tramite appositi software, è possibile generare delle foto geometricamente corrette, le quali raffigurano lo stato reale del cantiere al momento del volo.

Per la raccolta dei dati durante questo lavoro sono stati impiegati droni «senseFly eBee» [1] (fig. 1.1) ad ala fissa e prodotti dall'azienda «AgEagle».

Tra i droni in commercio, si cita anche il modello concorrente «MINI 2» [2] (fig. 1.2) dell'azienda «DJI»; la particolarità è il suo peso di 249g che consente la guida del velivolo senza bisogno di attestato ENAC [3] in quanto facente parte delle eccezioni riguardanti la *sottocategoria A1* limitata ai droni con peso sotto 250g o con marcatura C0. Inoltre, il MINI 2 è integrabile con il sistema «Litchi» [4] tramite un'apposita API. Si parla di un software di terze parti che gira su droni DJI e tra le tante possibilità che il software offre, permette voli con l'esecuzione di tragitti pre-programmati.
In ultimo, una nota di riguardo sul rapporto qualità-prezzo che lo rende un ottimo candidato per le imprese che lavorano in questo settore. Infatti, il DJI MINI 2 ha un costo che si aggira sui 500€, mentre un drone senseFly eBee di 10.000€ all'incirca.

In particolare, ciò che si discuterà in questa trattazione sarà il processo successivo alla raccolta dei dati tramite droni; esso produrrà dei modelli in grado di

Figura 1.1: Drone senseFly ebee

Figura 1.2: Drone DJI MINI 2

Figura 1.3: Dispositivo di comando del drone DJI MINI 2 con un dispositivo mobile come periferica

fare predizioni su futuri voli in un cantiere grazie ad algoritmi di Deep Learning (DL) [5]. A tal proposito, si creeranno reti neurali artificiali (ANN) [6] che sono un sottoinsieme del Machine Learning (ML) [7], oltre che l'elemento centrale degli algoritmi di DL, al fine di individuare lo stato avanzamento lavori in un cantiere, partendo appunto dai dati raccolti.

Il prodotto finale del processamento dei dati e dell'estrazione delle features sarà un'immagine del cantiere dove sono distinte per colore le zone relative a scavo aperto, scavo chiuso, e posa del tubo all'interno del cantiere.

I dati raccolti sul campo e gli output prodotti dai metodi di elaborazione sono integrati in una piattaforma digitale con funzionalità di repository, management, reporting delle attività sui cantieri, in ambiente georeferenziato, creando così un sistema ampio e generale che ingloba i vari passi, schematicamente mostrati in fig.1.4.

In questa maniera, le aziende del campo oli & gas possiedono un controllo automatizzato delle varie fasi di lavorazione, migliorando anche il rapporto con l'ente richiedente il lavoro, come può essere un ente pubblico.

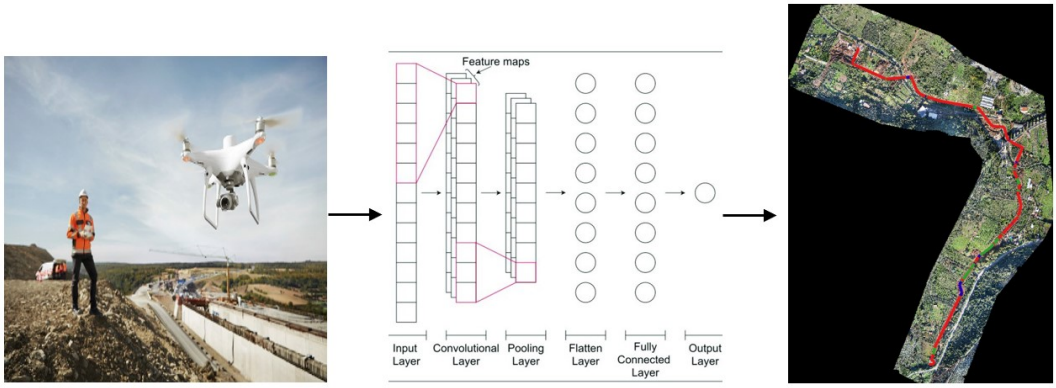


Figura 1.4: Processo generale del riconoscimento dello stato avanzamento lavori:
1) volo drone
2) elaborazione con modello DL
3) risultati grafici del modello

1.1 Stato dell'arte

Di seguito si citano le fonti riguardanti due progetti che hanno fatto uso di tecnologie quali i droni oltre ad algoritmi di DL e se ne descrivono gli obiettivi principali.

Il primo lavoro [8] si concentra sull'impiego sempre maggiore dei droni all'interno delle smart cities. Partendo da compiti già ricoperti, quali il controllo del traffico e delle costruzioni, fino ad arrivare ad altri in evoluzione quali il monitoraggio dei cambiamenti visuali nelle immagini raccolte durante un lungo periodo, con lo scopo di segnalare fenomeni di appropriazione illecita di suolo pubblico.
Così come avviene per i cantieri oli & gas, anche in questo caso, l'accertamento dello stato corrente di un'area di interesse è un compito tedioso ad oggi svolto manualmente da adetti qualificati. Perciò, entrano in gioco tecnologie come i droni, che riducono i tempi di percorrenza di grandi aree, garantendo spesso una maggiore precisione nella cattura dei dati. Con l'aiuto di un'architettura di DL che utilizzi Convolutional Neural Networks (CNN) è quindi possibile localizzare e identificare cambiamenti tra le immagini raccolte e quelle attuali o, come si illustrerà nella trattazione, tra differenti modelli digitali del terreno.

Invece, il secondo lavoro [9], sviluppa un metodo per il rilevamento e monitoraggio automatico di strade in costruzione attraverso immagini raccolte periodicamente nel tempo. Un modello di CNN è stato allenato con differenti immagini raccolte nel tempo grazie ad un drone eBee plus. Si noti che, anche in questa trattazione, si implementeranno modelli di CNN per cui il dataset utilizzato sarà il risultato dell'unione di dati raccolti in fasce temporali differenti, con l'intento di ottenere maggiore elasticità di classificazione dei modelli.

1.2 Obiettivo del lavoro

A partire da informazioni georeferenziate su un cantiere, e attraverso un algoritmo di DL, si automatizzerà il riconoscimento dello stato avanzamento lavori all'interno di un cantiere. In dettaglio, per un generico cantiere, si amplierà la capacità dell'algoritmo [10] di classificare le zone di

- scavo chiuso (SC);
- scavo aperto (SA);

con l'introduzione di una nuova feature:

- posa del tubo (PT).

Questi tasks incrementativi sono parte della totalità delle features di interesse per un'azienda del settore. In particolar modo, si parlerà dell'introduzione della PT, che è il punto chiave per passare da un problema di classificazione binaria ad uno con tre classi; da qui in poi l'estensione ad un numero maggiore di classi viene semplificata notevolmente poiché già presenti tecniche di analisi multi-classe. Infatti, come si potrà osservare, sono state implementate funzionalità che permettono una divisione generica e quanto più possibile eterogenea dei dati.

A seguire, in questa trattazione, si farà riferimento a due differenti cantieri presi in analisi che, per motivi di riservatezza, saranno menzionati come: cantiere in Zona A e cantiere in Zona B.

Capitolo 2

Materiali

In questo secondo capitolo, si introducono e commentano in dettaglio i dati di input e output, oltre che i software utilizzati per la realizzazione della fase di DL all'interno del framework più generale.

Per quanto riguarda i dati di partenza, si parlerà di come questi vengono trattati nelle fasi di pre-processamento al fine di raffinarli per i modelli delle reti neurali.

Le ANN sono quindi create in codice python all'interno della piattaforma Google Colab [11], e con un intenso uso delle librerie quali Keras [12], Sklearn [13].

Invece, per quanto riguarda la visualizzazione dei dati e anche dei risultati, si utilizza il software QGIS [14]. Infatti, oltre ad essere utile per identificare campioni dei dati reali su cui fare il training dei modelli, il software permette di associare l'output generato dalla ANN con i profili e le ortofoto di un particolare cantiere.

Infine, si parla di come l'output è generato e salvato in un file di testo appositamente formattato.

2.1 Dataset iniziale

Inizialmente è stato necessario lavorare con un'ortofoto [15] (illustrazione in fig. 2.1) del cantiere per generare un dataset con dati reali da poter utilizzare durante il training di un modello di ANN. Per creare un'ortofoto si scattano tante immagini sequenziali dell'area da riprodurre cercando di mantenere un'alta sovrapposizione tra due fotogrammi vicini. Le foto raccolte con i droni sono poi trattate con appositi software per generare una fotografia ortorettificata.

Figura 2.1: Ortofoto del cantiere in zona A, immagine presa da QGIS

Figura 2.2: DSM del cantiere in zona A in scala di grigi, immagine presa da QGIS

Di seguito, si impiegheranno i DSM [16] (illustrazione in fig. 2.2), ovvero modelli digitali della superficie che comprendono riferimenti geografici in coordinate oltre a dati di elevazione del terreno. La maniera in cui questi vengono generati è analoga al caso delle ortofoto.

Relativamente al dataset iniziale, nei cantieri della Zona A e B, la presenza di voli a disposizione per la creazione dei modelli e il test degli stessi è mostrata nei grafici 2.3 e 2.4.

2.2 Preprocessamento

La fase di pre-processamento mira a creare i profili dei DSM di uno o più voli scelti per il training di un modello della ANN.
Successivamente, nello stadio finale del pre-processamento, vengono esportati su un file di tipo JSON [17] i profili del DSM scelto insieme alle labels di ogni profilo, classificate manualmente. Questo file sarà l'input della ANN.

Creazione polilinea e relativi profili

Una prima parte ha riguardato l'osservazione del cantiere attraverso la relativa ortofoto RGB. Da quest'ultima vogliamo ricavare le zone di appartenenza delle tre classi. Per far ciò, è necessario creare una polilinea (una sequenza di segmenti di linea collegati come oggetto unico) che si estenda lungo tutta la linea dei lavori in corso (illustrazione in fig. 2.5). Dunque, utilizzando il software QGIS, si crea uno shapefile ESRI (Environmental Systems Research Institute) [18, 19], file con estensione SHP,

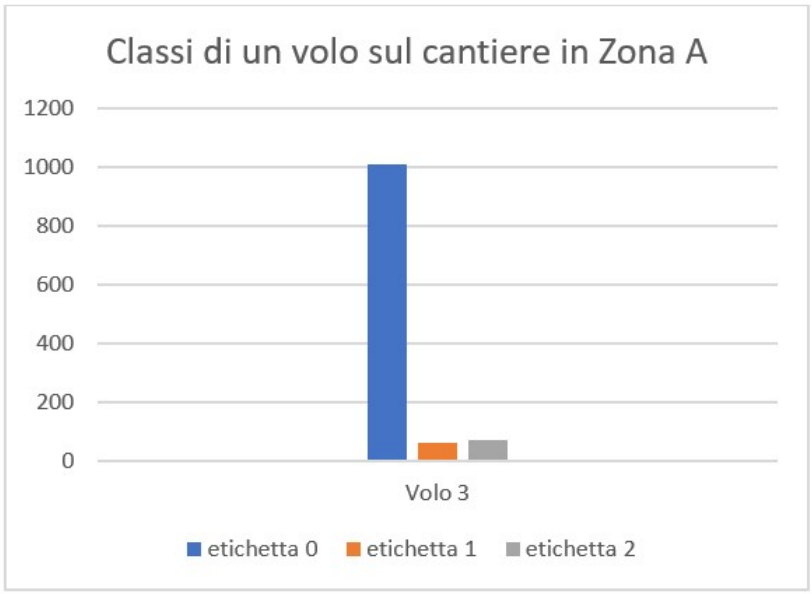


Figura 2.3: Distribuzioni delle classi di SA (label 0), di SC (label 1) e di PT (label 2) del cantiere in zona A

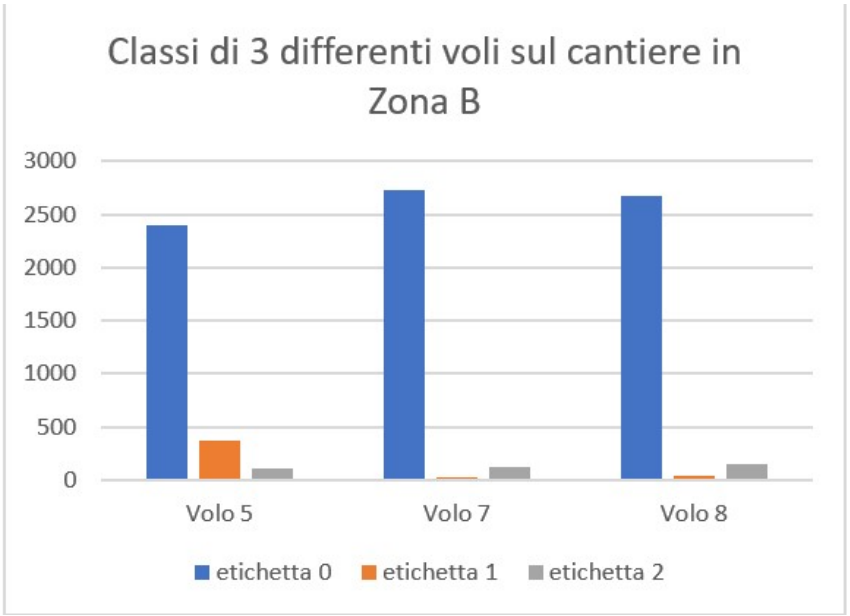


Figura 2.4: Distribuzioni delle classi di SA (label 0), di SC (label 1) e di PT (label 2) del cantiere in zona B

Figura 2.5: Shapefile del cantiere in zona A, immagine presa da QGIS

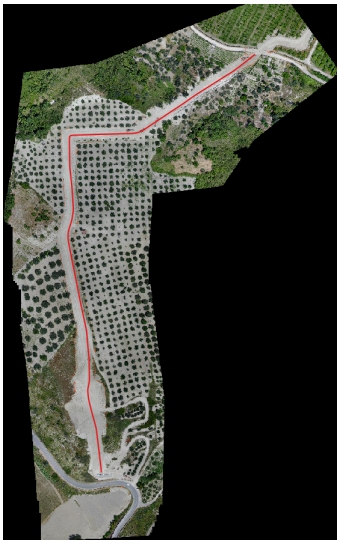


Figura 2.6: Smoothing sullo shapefile del cantiere in zona A, immagine presa da QGIS

la cui geometria è di tipo LineString. In pratica, si tratta di un vettore digitale contenente informazioni relative alla posizione e attributi ad essa correlati.

Inoltre, nella classe python *elevation-analysis.py* si lavora sullo shapefile effettuando operazioni di interpolazione dei segmenti che costituiscono la polilinea (vedi metodo *interpolate-shp2points(shp-path, n)*), oltre che di smoothing (vedi metodo *chaikins-smoothing(shp-path, refinements)*) per generare una curva (illustrazione in fig. 2.6).
Inoltre in dettaglio si espongono in fig. 2.7) le tre varie fasi dello shapefile.

Successivamente, tramite il metodo *create-ticks(shp-path, distance, tick-length)* della classe *chaintick.py* si generano i profili (illustrazione in fig. 2.8 e 2.9), ovvero segmenti perpendicolari alla polilinea, di lunghezza pari al parametro *tick-length* e distance pari a *distance*. In pratica, all'interno del metodo si crea un nuovo layer *out-ln-lyr* di geometria LineString, con sistema di riferimento spaziale uguale al layer dello shapefile in input, con l'aggiunta di due nuovi campi, sottoforma di colonne *"CHAINAGE"* e *"ID"*, per il nuovo layer di output.
In questo modo è possibile stabilire le zone di appartenenza di ciascuna classe tramite gli ID dei profili.

Figura 2.7: Shapefiles a confronto:
in *giallo* lo shapefile originale;
in *blu* successivamente all'interpolatizione;
in *rosso* dopo lo smoothing.

Figura 2.8: Profili del cantiere in zona A, immagine presa da QGIS

Figura 2.9: Dettaglio sui profili del cantiere in zona A, immagine presa da QGIS

Generazione file di input per la ANN

Proseguendo, tramite il metodo *export-dsmprofiles-dict(...)* della classe *elevation-analysis.py* si genera in output un file JSON denominato *"ElevationValues-full.json"*, il quale racchiude al suo interno tutte le informazioni di elevazione dei profili oltre a quelle di classificazione del generico volo X sul generico cantiere Z. Queste informazioni sono incapsulate dentro una lista di liste dove ognuna di queste contiene l'ID del profilo, i suoi valori di elevazione e il suo stato.

Tre sono gli stati ammissibili per un singolo profilo di un cantiere Z, e questi sono rappresentati nel file JSON attraverso le seguenti labels numeriche:

- SC label "0",
- SA label "1",
- PT label "2".

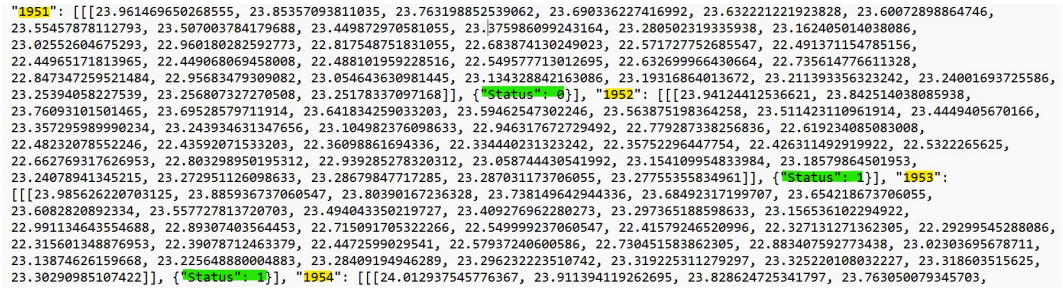
Si noti che, per aggiungere la label 2 relativa alla posa del tubo nel cantiere, è stato necessario creare una nuova chiave "*pdigdict=new-key: 2*" oltre a definire un ulteriore intervallo di appartenenza "*merged-range-pipe=chain(range(,))*" all'interno del metodo sopra menzionato. [1] L'intervallo stabilito è relativo ad uno specifico volo ed è utilizzato per generare il dateset di training del modello. [2]

Nella Figura 2.10, si mostra un esempio del contenuto del file JSON, dove in giallo è evidenziato l'ID di un profilo, mentre in verde la label a esso relativa.
Inoltre, in fase di estrazione di questi dati, si farà riferimento a:

- *profiles* come l'insieme dei valori di elevazione di ogni profilo e relativi alla prima colonna;
- *labels* come l'insieme degli stati di ogni profilo e relativi alla seconda colonna.

[1] Per il training di un nuovo modello di ANN con dati di uno o più voli non previamente analizzati, si modificheranno i paramentri *merged-range* e *merged-range-pipe* con differenti range relativi alle labels 1 e 2 per ciascun nuovo volo.

[2] Nella classe *elevation-analysis.py* è necessario cambiare i percorsi alle cartelle sul proprio dispositivo per eseguire correttamente il codice



"1951": [[[23.961469650268555, 23.85357093811035, 23.763198852539062, 23.690336227416992, 23.632221221923828, 23.60072898864746,
23.55457878112793, 23.507003784179688, 23.449872970581055, 23.375986099243164, 23.280502319335938, 23.162405014038086,
23.02552604675293, 22.960180282592773, 22.817548751831055, 22.683874130249023, 22.571727752685547, 22.491371154785156,
22.44965171813965, 22.449068069458008, 22.488101959228516, 22.549577713012695, 22.632699966430664, 22.735614776611328,
22.847347259521484, 22.95683479309082, 23.054643630981445, 23.134328842163086, 23.19316864013672, 23.211393356323242, 23.24001693725586,
23.25394058227539, 23.256807327270508, 23.25178337097168]], {"Status": 0}], "1952": [[[23.94124412536621, 23.842514038085938,
23.76093101501465, 23.69528579711914, 23.641834259033203, 23.59462547302246, 23.563875198364258, 23.51142311096191, 23.4449405670166,
23.357295989990234, 23.243934631347656, 23.104982376098633, 22.946317672729492, 22.779287338256836, 22.619234085083008,
22.48232078552246, 22.43592071533203, 22.36098861694336, 22.334440231323242, 22.35752296447754, 22.426311492919922, 22.5322265625,
22.662769317626953, 22.803298950195312, 22.939285278320312, 23.058744430541992, 23.154109954833984, 23.18579864501953,
23.24078941345215, 23.272951126098633, 23.28679847717285, 23.287031173706055, 23.27755355834961]], {"Status": 1}], "1953":
[[[23.985626220703125, 23.885936737060547, 23.80390167236328, 23.738149642944336, 23.68492317199707, 23.654218673706055,
23.6082820892334, 23.557727813720703, 23.494043350219727, 23.409276962280273, 23.297365188598633, 23.156536102294922,
22.991134643554688, 22.89307403564453, 22.715091705322266, 22.549999237060547, 22.41579246520996, 22.327131271362305, 22.29299545288086,
22.315601348876953, 22.39078712463379, 22.4472599029541, 22.57937240600586, 22.730451583862305, 22.883407592773438, 23.02303695678711,
23.13874626159668, 23.225648880004883, 23.28409194946289, 23.296232223510742, 23.319225311279297, 23.325220108032227, 23.318603515625,
23.30290985107422]], {"Status": 1}], "1954": [[[24.012937545776367, 23.911394119262695, 23.828624725341797, 23.763050079345703,

Figura 2.10: Esempio di alcune righe del file *"ElevationValues-full.json"*

Normalizzazione del dataset

La terza e ultima parte del pre-processamento si svolge direttamente all'interno di un notebook di Google Colab e coinvolge i dati ricavati dal file JSON sopra menzionato; questa si occupa della normalizzazione dei dati raccolti.
L'obiettivo è quello di ottenere una scala uniforme che vada da 0 alla lunghezza massima tra le lunghezze dei profili contenuti nel JSON, ed evitare così discrepanze nell'analisi degli stessi.
In particolare, si inizializza un array di zeri, chiamato *full-data*, e che possiede:

- numero di righe dato da n-data=len(profiles), ovvero pari al numero di elementi contenuti nella lista di profili *profiles*;
- numero di colonne pari a n-steps=42, ovvero un valore intermedio fra quelli delle lunghezze dei singoli profili creati.

All'interno di un ciclo *for* che scansiona tutti gli elementi contenuti in *profiles*, si crea una sottolista *sub-list*, e mediante costrutti condizionali si determina la lunghezza limite dei profili. Infine, al termine del ciclo, in *full-data* si sovrascrivono uno ad uno i profili normalizzati a scala comune e indicizzati mediante una variabile contatore *row* che incrementa ad ogni ciclo. Dentro la variabile *full-data* saranno quindi contenuti profili con affiancati i rispettivi valori di elevazione.

2.3 Dataset finale

L'ultimo anello di questo processo di elaborazione è la generazione di un file .txt. Questo file contiene al suo interno i profili di un cantiere distinti per ID, assieme alle predizioni che un particolare modello ha effettuato relativamente ad un volo (esempio in fig. Figura 2.11). Il file di testo è organizzato in due colonne: la prima da sinistra indica l'*ID* di ciascun profilo, mentre la seconda indica la *label* predetta dal modello per quel profilo.

Figura 2.11: Esempio del file di testo contente risultati delle predizioni

2.4 Software utilizzati

In questo paragrafo, si illustrano i software utilizzati per la realizzazione della parte AI all'interno del framework finale.

2.4.1 Software QGIS

QGIS è un software GIS (Geographic Information System) di informazione geografica open-source facile da usare, rilasciato sotto la GNU General Public License [20], che permette di visualizzare, organizzare, analizzare e rappresentare dati spaziali.

Nella zona di sinistra, in fig. 2.12, è possibile utilizzare il panello *Browser Panel* per navigare tra i file che verranno aperti nella zona *Layers Panel*. Attraverso quest'ultima è possibile selezionare i layers che si desidera visualizzare in *Map*.

Di rilevanza, la capacità del software di supportare vari formati differenti di dati e di mostrare layers sovrapposti.

In particolare, accedendo alle proprietà di ciascun layer con il tasto destro, è possibile aggiungere informazioni ad esso mediante operazioni di *join* o di categorizzazione dei dati. Queste saranno utili per graficare in QGIS i risultati ottenuti dalle predizioni dei modelli che si andranno a creare.

Funziona su Linux, Unix, Mac OSX, Windows e Android e supporta numerosi formati vettoriali, raster e database.

2.4.2 Google Colab

Si utilizza la piattaforma Google Colab collegata con Google Drive [21] per la gestione dei dati, oltre al linguaggio di programmazione python, versione 3.8.10 [22].

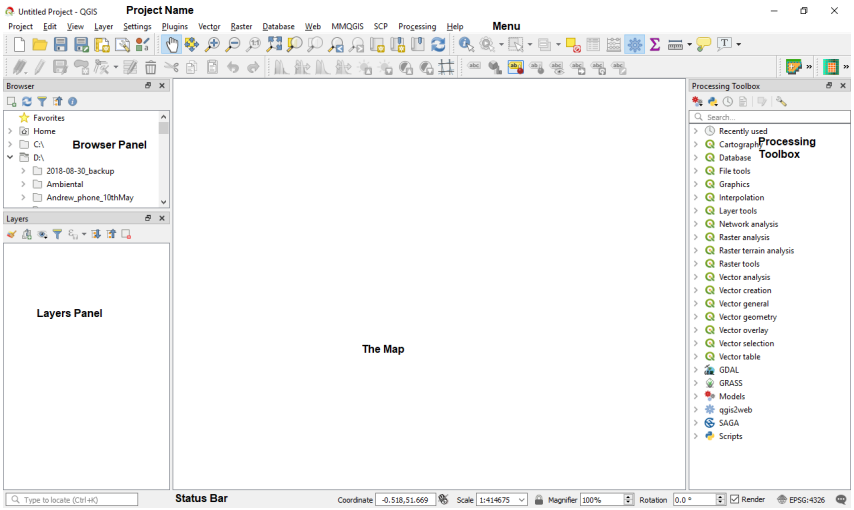


Figura 2.12: QGIS guida di avvio veloce

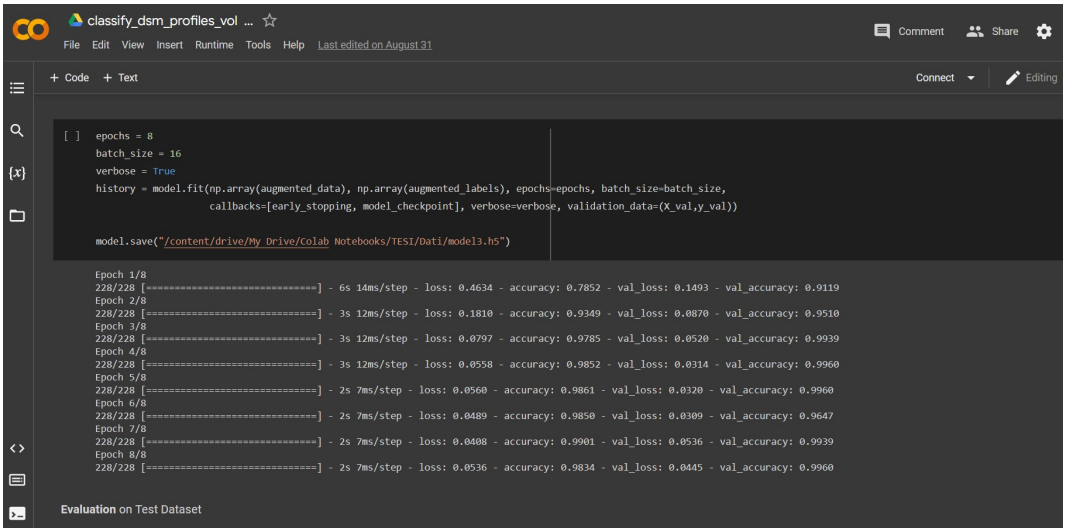


Figura 2.13: Google Colab - screenshot training di un modello

Questo tool gratuito permette di eseguire codice python direttamente dal browser, sfruttando le capacità di calcolo dei server Google.

Capitolo 3

Metodi

3.1 Task applicativo

L'obiettivo è quello di soddisfare un task multiclasse: predire l'appartenenza di un profilo alle tre classi (SC, SA, PT). Di conseguenza, è stata modificata l'accuracy del modello da "binary cross entropy" a "sparse categorical cross entropy" poiché le labels sono numeri interi: 0, 1, 2.

In fig. 3.1 e 3.2 si mostra il diagramma di flusso dell'intera operazione, dal principio alla fine.

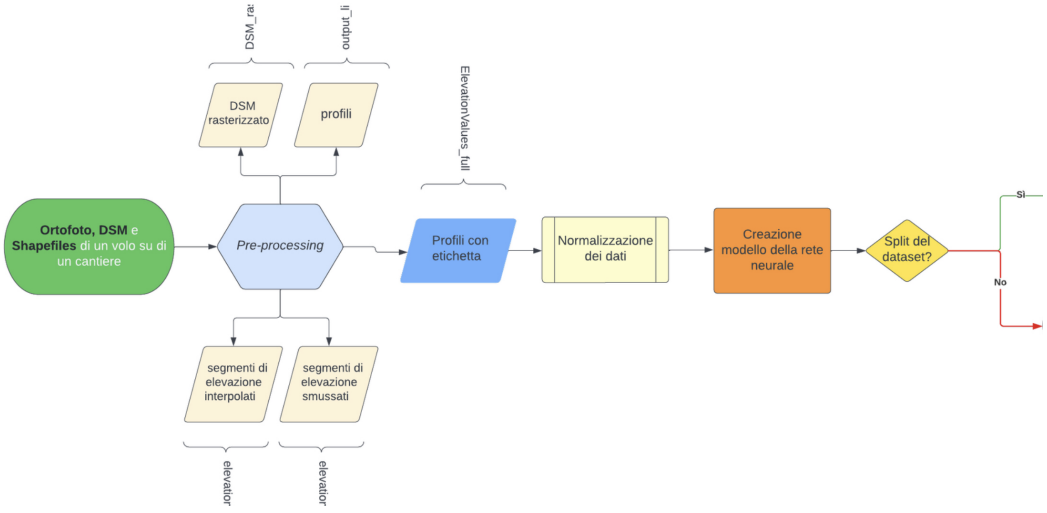


Figura 3.1: Diagramma di flusso per la creazione dei modelli ANN, parte 1

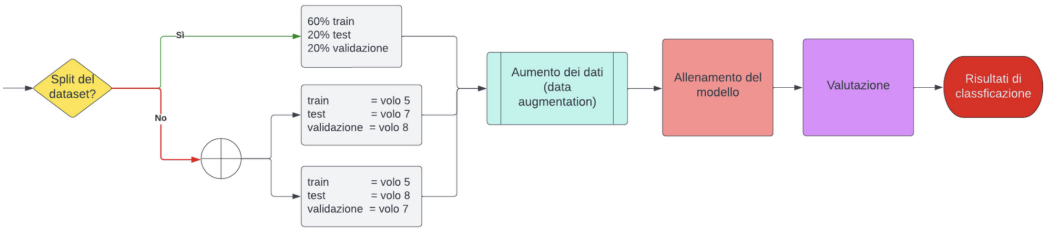


Figura 3.2: Diagramma di flusso per la creazione dei modelli ANN, parte 2

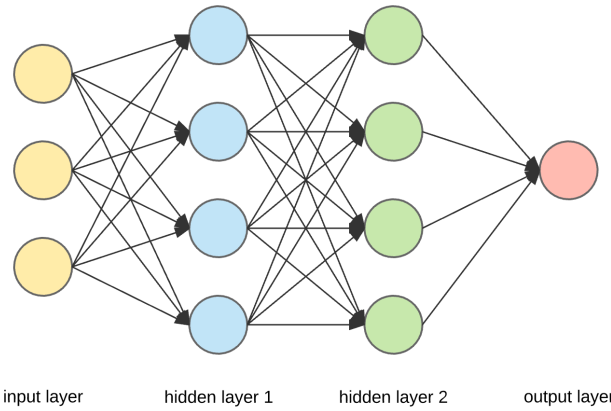


Figura 3.3: Esempio di una generica rete neurale multi-layer

3.2 Metodologia di Deep Learning

Le ANN sono composte da livelli di nodi con un livello di input, uno o più livelli nascosti e un livello di output (vedi fig. 3.3).
In particolare, ciascun nodo, o neurone artificiale, si connette ad un altro e ha un peso e una soglia associati. Se l'output di un singolo nodo qualsiasi è al di sopra del valore di soglia specificato, tale nodo viene attivato, inviando i dati al successivo livello della rete. In caso contrario, non viene passato alcun dato al livello successivo della rete.

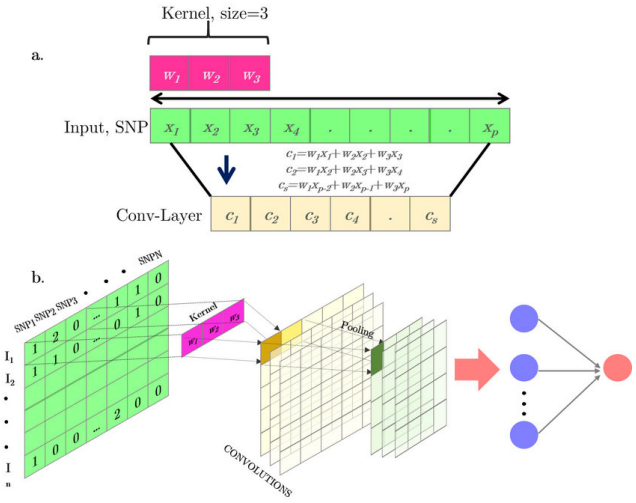


Figura 3.4: Semplice schema di un'operazione convoluzionale 1D

Nell'ambito del ML, ed in particolare del DL, esistono tipi diversi di ANN. In questa sperimentazione, in particolare, si utilizzeranno delle CNN. Queste reti neurali prendono il nome dal fatto che utilizzano degli strati convoluzionali che spesso sono utilizzati per compiti di classificazione e di computer vision. In generale, una CNN è progettata per apprendere automaticamente le gerarchie spaziali delle caratteristiche ricercate. Essa comprende tipicamente tre tipi di livelli, noti anche come blocchi: convoluzione, pool e livelli completamente connessi. I layers di convoluzione e pooling eseguono l'estrazione delle features desiderate, che poi vengono mappate nell'output finale dal layer fully-connected. In particolare, è stato dimostrato che le CNN 1D [23, 24] sono relativamente più facili da addestrare e offrono la minima complessità computazionale, raggiungendo livelli di prestazioni all'avanguardia. Dato che si lavora con dati in input mono-dimensionali, e non direttamente con immagini, è conveniente l'utilizzo di layers 1D (vedi fig. 3.4) per la creazione del nostro modello.

Di seguito, nel diagramma sottostante, si riporta la struttura a strati del modello sequenziale. In ordine di apparizione, si hanno:

- 4 layers convoluzionali
- 1 layer di average pooling
- 2 layers fully-connected

In particolare, si utilizza un layer convoluzionale 1D input con una dimensione del filtro più grande rispetto ai successivi al fine di catturare un maggior numero di caratteristiche iniziali. Successivamente, si è utilizzato un layer di Average Pooling, che effettua l'estrazione dei valori medi da un raggruppamento di dimensione specifica, poiché questo produce migliori risultati empirici rispetto ad un classico layer di Max Pooling. Infine, due layers di neuroni fully-connected in cui il primo utilizza come funzione di attivazione una reLu [25] (Rectified Linear Activation Function) mentre

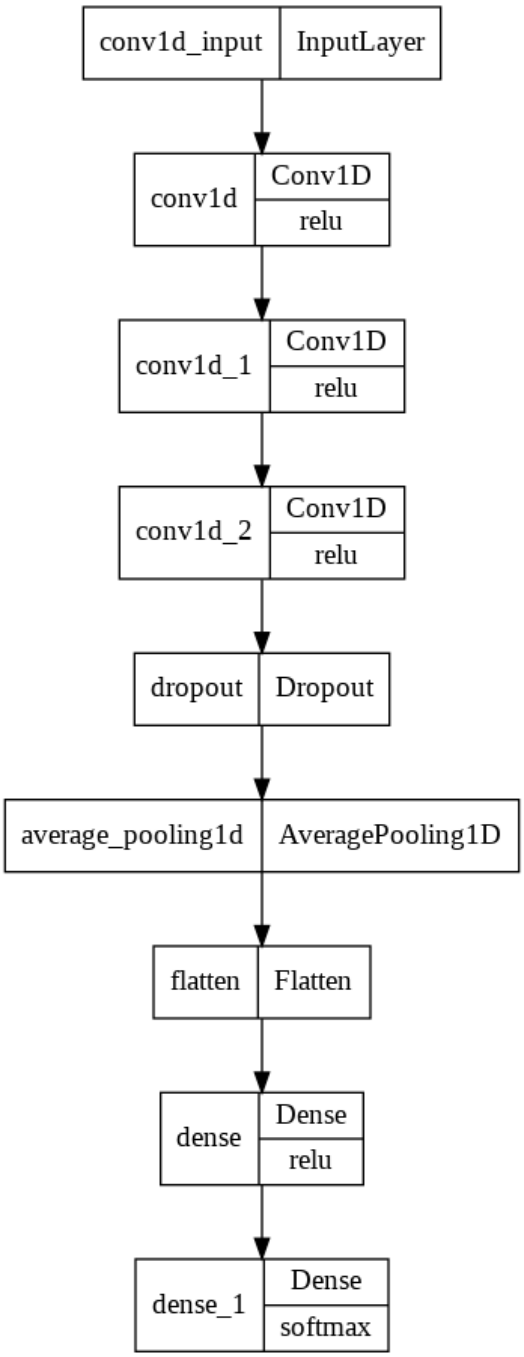


Figura 3.5: Modello sequenziale della nostra rete neurale con strati convolutivi

la seguente ed ultima utilizza una softmax che si occupa di convertire un vettore di K numeri reali in una distribuzione di probabilità con K possibili uscite.

Si noti che nel diagramma i layers di Dropout e Flatten non sono stati menzionati nella discussione perché, seppur funzionali alla realizzazione del modello, non influiscono sullo stesso.

3.3 Data augmentation

Tra la creazione e il training del modello si utilizzano tecniche di data augmentation. L'obiettivo è quello di generare nuovi campioni delle classi minoritarie e avere così un modello che si alleni su classi più equamente distribuite per predizioni più accurate. In particolare, si utilizzano:

- lo SMOTE [26], ossia una tecnica di sovra-campionamento sintetico delle minoranze che consiste in sintetizzare nuovi esempi di classi minoritarie. Questi nuovi esempi vengono creati selezionando elementi vicini in intervalli omogenei e disegnando una linea tra gli elementi dell'intervallo per aggiungere un nuovo campione in un punto lungo quella linea.

- Il flipping del dataset di training, che consiste nell'invertire l'ordine dei profili e delle labels rispetto ad un asse verticale, raddoppiando i dati a disposizione per il training del modello.

3.4 Funzione di perdita (loss) e metriche

L'algoritmo di ottimizzazione utilizzato è Adam [27], un'estensione del metodo iterativo di Stochastic Gradient Descent (SGD) per l'aggiornamento dei pesi del modello in relazione al dataset di training dello stesso. Adam unisce i vantaggi di altri due algoritmi di SGD: AdaGrad e RMSProp. In dettaglio, esso facilita il calcolo dei learning rates per ciascun parametro, utilizzando il primo e il secondo momento del gradiente.

Inoltre, come metrica nella fase di compilazione del modello, si utilizza una classe personalizzata: «BalancedSparseCategoricalAccuracy». Questa setta i pesi di ogni classe in maniera inversamente proporzionale alla numerosità delle labels nelle singole classi; in tal modo ottempera parzialmente alla sproporzione tra le classi del dataset in fase di ricerca di pesi ottimali per il modello.

3.5 Ottimizzazione iperparametri

Importante per il training del modello di una ANN è la scelta degli iperparametri, quali il numero di epoche e della batch size.

Il primo definisce il numero di volte che la rete neurale riceverà l'intero dataset di training, mentre il secondo indica il numero di campioni da utilizzare per l'aggiornamento del gradiente ad ogni iterazione.

Si è visto che, generalmente, un valore della batch-size pari a 16 con numero di epoche pari a 24 è soddisfacente per il training su un dataset ricavato dallo split di uno stesso volo. Invece, con un dataset ricavato da voli differenti, quindi con un numero totale di campioni superiore, e con una maggiore diversificazione dei dati, è conveniente utilizzare un valore della batch-size inferiore e pari a 8 con numero di epoche uguale a 30.

In aggiunta, sono state integrate due classi di callback nel modello:

- *EarlyStopping* [28] che interrompe il training del modello nel momento in cui il valore della funzione di perdita del dataset di validazione (*val-loss*) inizia a crescere nuovamente; l'indice di tolleranza è il parametro *patience*. Ciò significa che, dato *patience=3*, il training si interrompe prima del numero di epoche stabilite se i valori della *val-loss* continuano ad aumentare dopo 3 epoche consecutive. In tal modo si limita l'overfitting del modello e si riduce il numero di epoche necessarie.

- *ModelCheckpoint* [29] che salva i pesi del modello durante il training. Inoltre, tramite il parametro *save-best-only=True* si è specificato di salvare solo i pesi migliori all'interno di un file esterno. L'obiettivo è dunque quello di ricaricare questi pesi e utilizzarli per le successive predizioni.

3.6 Dataset splitting

Tra i concetti alla base del DL si ha quanto segue: più dati si utilizzano per il training di un modello di una ANN, maggiori probabilità ci sono che esso "impari dagli errori". Allo stesso modo, quante più deduzioni il modello può estrarre dai dati, tanto più lo stesso sarà preciso nell'identificare correttamente le classi. Ad ogni modo, in fase di training è utile stimare la bontà di un modello creato per poterne modificare gli iperparametri in tempo reale e ottenere migliori prestazioni. Per fare ciò, in presenza di una ridotta quantità di dati, si rende necessario dividere il dataset di partenza in due. Il fine è quello di produrre due dataset differenti: uno di *training* del modello e l'altro di *validazione* dello stesso.

Inoltre, si osservi che se si usassero tutti i dati a disposizione per il training e per la validazione di un modello, non resterebbero altri dati sconosciuti al modello e su cui valutare in maniera imparziale le sue prestazioni.
È qui che nasce l'esigenza di effettuare una ulteriore suddivisione del dataset; bisogna conservare una parte del dataset (a cui il modello non potrà accedere in fase di training) per effettuare un test sul modello.

In ultima analisi, bisogna disporre di 3 dataset distinti per implementare delle ANN: uno per allenare, uno per validare e uno per testare il modello della rete.

3.7 Procedure sperimentali

Secondo quanto discusso in *dataset splitting (3.6)* e per valutare la capacità di generalizzazione dei modelli, sono stati individuati diversi split dei dataset, racchiusi all'interno di 3 principali ambiti di sperimentazione:

- generalizzazione su uno stesso volo;
- generalizzazione su un altro volo dello stesso cantiere;
- generalizzazione sul volo di un altro cantiere.

Si noti che per generalizzione si intende la capacità che il modello della ANN ha di classificare i profili di un volo rispetto a un determinato fattore (uno stesso volo, un altro volo, un altro cantiere).

Tabella 3.1: Schema sintetico delle fasi di sviluppo di un modello

START:	- ortofoto RGB, - DSM file singolo volo,
PRE-PROCESS	- creazione shapefiles relativi al cantiere di quel volo, - ottimizzazione dataset per il training del modello, - scrittura file JSON contenente profili con labels reali.
PROCESS:	- creazione modello.
END:	- generazione file .txt contenente ID profili con labels predette.

3.7.1 Esperimento 1 (Generalizzazione sullo stesso volo)

In questo primo esperimento si fanno predizioni per il volo n.3 sul cantiere A, con split del dataset come segue:

- 60% per il training;
- 20% per la validation;
- 20% per il test.

Durante la fase di split si è aggiunto esplicitamente il parametro *stratify* con valore *labels*. Quest'accortezza permette di mantenere le proporzioni iniziali delle tre classi anche nei tre sottoinsiemi di training, validation e test. Infatti, è necessario fare ciò perché altrimenti la divisione del dataset sarebbe randomica per default tramite il metodo di *train-test-split*. In ordine, prima si divide il dataset nei sottoinsiemi di

training e di test, poi in training e validation (quest'ordine non influisce sul risultato finale).

Si riportano sinteticamente in tabella alcuni parametri e iperparametri impiegati per il training del modello 3 sul dataset del volo 3, Zona A:

Tabella 3.2: Dati relativi al training del modello3 sul volo 3, cantiere Zona A

FEATURES	VALUES
profiles	1141 profili totali
labels	label 0: 1012 label 1: 59 label 2: 70
metrics	Balanced Sparse Categorical Accuracy
callbacks	EarlyStopping, ModelCheckpoint
Split original dataset	60% train 20% validation 20% test stratify=labels
Data augmentation	imbalanced dataset correction; increase dataset size flipping training profiles and labels
epoche	15
batch size	16

3.7.2 Esperimento 2 (Generalizzazione sullo stesso volo)

In questo secondo esperimento si fanno predizioni per il volo n.5 sul cantiere B, con split del dataset come segue:

- 60% per il training ;
- 20% per la validation;
- 20% per il test.

Analogamente al caso precedente durante la fase di split si è aggiunto esplicitamente il parametro stratify con valore labels.

Si riportano sinteticamente in tabella alcuni parametri e iperparametri impiegati per il training del modello 5 sul dataset del volo 5, Zona B:

Tabella 3.3: Dati relativi al training del modello5 sul volo 5, Zona B

FEATURES	VALUES
profiles	2869 profili totali
labels	label 0: 2403 label 1: 364 label 2: 102
metrics	Balanced Sparse Categorical Accuracy
callbacks	EarlyStopping, ModelCheckpoint
Split original dataset	60% train 20% validation 20% test stratify=labels
Data augmentation	imbalanced dataset correction; increase dataset size flipping training profiles and labels
epoche	17
batch size	16

3.7.3 Esperimento 3 (Generalizzazione su un altro volo)

In questo terzo esperimento per il volo 5 sul cantiere B, si cambia lo split del dataset come segue:

- volo5, cantiere B, per il training;
- volo 8, cantiere B, per la validation;
- volo 7, cantiere B, per il test.

Si vuole generalizzare il caso precedente testando il modello su un altro volo dello stesso cantiere.
Si riportano sinteticamente in tabella alcuni parametri e iperparametri impiegati per il training del modello 5-8-7 sul dataset del volo 5, Zona B:

Tabella 3.4: Dati relativi al training del modello 5-8-7 sul volo 5, Zona B, con 3 voli distinti

FEATURES	VALUES
profiles	8607 profili totali
metrics	Balanced Sparse Categorical Accuracy
callbacks	EarlyStopping, ModelCheckpoint
Split original dataset	volo 5 train volo 8 validation volo 7 test
Data augmentation	imbalanced dataset correction; increase dataset size flipping training profiles and labels
epoche	5
batch size	8

3.7.4 Esperimento 4 (Generalizzazione su un altro volo)

Come fatto nel precedente caso per il volo 5 sul cantiere B, si modifica lo split del dataset come segue:

- volo5, cantiere B, per il training;
- volo 7, cantiere B, per la validation;
- volo 8, cantiere B, per il test.

Si vuole generalizzare il caso precedente testando il modello su un altro volo dello stesso cantiere.
Si riportano sinteticamente in tabella alcuni parametri e iperparametri impiegati per il training del modello 5-7-8 sul dataset del volo 5, Zona B:

Tabella 3.5: Dati relativi al training del modello 5-7-8 sul volo 5, Zona B, con 3 voli distinti

FEATURES	VALUES
profiles	8607 profili totali
metrics	Balanced Sparse Categorical Accuracy
callbacks	EarlyStopping, ModelCheckpoint
Split original dataset	volo 5 train volo 7 validation volo 8 test
Data augmentation	imbalanced dataset correction; increase dataset size flipping training profiles and labels
epoche	5
batch size	8

3.7.5 Esperimento 5 (Generalizzazione su un altro cantiere)

A questo punto viene effettuata una prima prova su di un caso reale in cui si utilizza un modello pre-allenato su un cantiere per fare predizioni su un volo relativo ad un altro cantiere.
In particolare, si testa il modello 5 sul volo 3 del cantiere A. Il modello 5, creato sul cantiere B, presenta una divisione del dataset come segue:

- 60% per il training;
- 20% per la validation;
- 20% per il test.

Si noti che in questo particolare caso una parte del dataset, quella di test, non verrà impiegata poiché si testa sul volo 3 del cantiere A. Si è comunque mantenuta questa divisione del dataset per il modello 5 perché è stato previamente creato e salvato (Esperimento 3.7.2).

3.7.6 Esperimento 6 (Generalizzazione su un altro cantiere)

Si effettua una seconda prova su di un caso reale in cui si utilizza un modello allenato su un cantiere per fare predizioni su un volo relativo ad un altro cantiere. In particolare, si testa sullo stesso volo 3 del cantiere A, ma con il modello 5-8-7. Nel modello 5-8-7 i dati sono divisi come segue:

- volo5, cantiere B, per il training;

- volo 8, cantiere B, per la validation;
- volo 7, cantiere B, per il test.

Si noti che anche in quest'altro caso una parte del dataset, quella di test e relativa al volo 7, non verrà impiegata poiché si testa sul volo 3 del cantiere A. Si è comunque mantenuta questa divisione del dataset per il modello 5-8-7 perché è stato previamente creato e salvato (Esperimento 3.7.3).

3.7.7 Esperimento 7 (Generalizzazione su un altro cantiere)

Si effettua una terza e ultima prova su un caso reale in cui si utilizza un modello allenato su un cantiere per fare predizioni su un volo relativo ad un altro cantiere. In particolare, si testa sullo stesso volo 3 del cantiere A però con il modello 5-8. Nel modello 5-8 si uniscono i dati di 2 voli (voli 5 e 8, Zona B) per accrescere il dataset di training; inoltre si utilizza il dataset del volo 7, Zona B, per validare il modello così creato.

3.8 Osservazioni

1. I dati ottenuti in output dal metodo «model.predict(np.array(X-test))» saranno vettori di numeri che rappresentano la probabilità di appartenere o meno ad una delle tre classi per ogni profilo estratto. I valori variano tra 0 e 1 dove 1 indica maggiore probabilità per un profilo di appartenere a quella classe.

2. Si osservi che, un fattore che ha influenzato negativamente il training del singolo modello è stato l'eterogenea distribuzione delle tre classi (SA, SC, PT). Come è facile notare grazie al grafico in fig. **??**, c'è una netta sproporzione tra la grande copiosità della label 0, e la ridotta presenza delle altre due labels. Per ovviare agli sbilanci, sono state implementate varie tecniche di cui si è parlato nei paragrafi 3.3 e 3.4, che hanno contribuito a livellare questo gap.

3. Inoltre, è bene evidenziare che, durante la fase di preprocessamento (sezione 2.2), sono presenti alcune difficoltà nello stabilire con precisione caratteristiche di determinati intervalli di profili. Si osserva che queste sono dovute anche alla presenza di mezzi di trasporto lungo il tracciato del cantiere, a zone fortemente oscurate ed ad eventuali interruzioni percorso (ferrovie, ponti, etc.).

4. Infine, si noti che le classi reali di appartenenza dei profili (SC, SA o PT), di cui il modello effettuerà le predizioni, sono soggette a errore umano di classificazione dei casi reali. Infatti preventivamente al training è compito di una persona stabilisce intuitivamente gli intervalli dei profili osservando l'ortofoto del volo di interesse. Ciò chiaramente comporta possibili errori nelle previsioni finali.

Capitolo 4

Risultati

Riguardo i risultati ottenuti dal training sui modelli delle ANN, i due parametri da mettere in risalto sono:

1. l'accuratezza dello stesso nell'identificare i task prestabiliti;

2. la funzione di perdita tra etichette reali e quelle stimate sui profili in analisi.

Nei prossimi passi si illustreranno in dettaglio gli esperimenti svolti, evidenziandone le curve di training dei modelli impiegati, così come le curve delle funzioni di perdita. Inoltre, si presentano le matrici di confusione e i report sulle classificazioni effettuate per una chiara visualizzazione dei risultati. In dettaglio, si descrivono brevemente le metriche utilizzate nei report [30]:

- $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ è il rapporto tra le previsioni corrette e il numero totale di previsioni; valuta l'efficacia complessiva di un classificatore (da massimizzare).

- $precision = \frac{TP}{TP+FP}$, è il numero di previsioni positive corrette; indica la corrispondenza delle labels con le features positive fornite al classificatore (da massimizzare).

- $recall = \frac{TP}{TP+FN}$ è il numero di previsioni positive esatte diviso per il numero totale di previsioni corrette; indica l'efficacia di un classificatore nell'identificare tutte le labels esatte (da massimizzare).

- $f1score = 2 \cdot \frac{precision \cdot recall}{precision+recall}$ è la media armonica dei valori di *precison* e *recall*; indica il rendimento combinato delle due metriche (da massimizzare).

- *macro avg* è la media aritmetica dei valori di tutte le classi per ciascuna metrica (precision, recall, f1-score);

- *weighted avg* è analoga a *macro avg*, ma tiene conto del numero reale di occorrenze della particolare classe nel dataset. [1]

[1] TP: true positive, TN: true negative, FP: false positive, FN: false negative

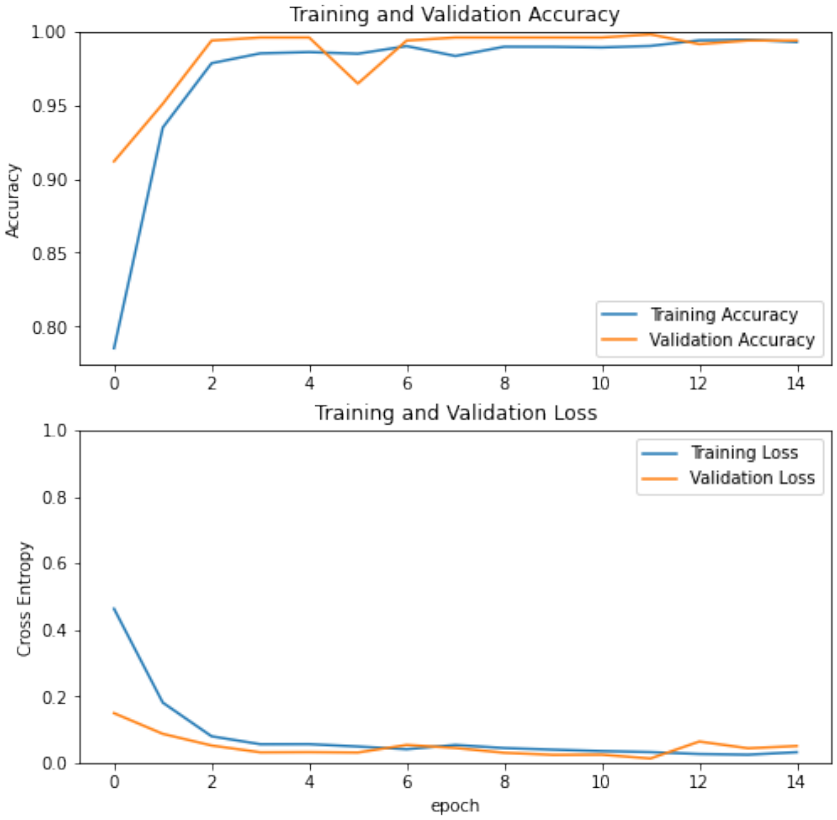


Figura 4.1: Curve di training e della funzione di perdita relative modello 3 allenato sul volo 3, Zona A

4.1 Esperimento 1

Il training è stato svolto sul volo 3, Zona A, con i parametri riportati in Tabella 3.2. Si riportano dunque le curve del training del modello creato con 1D CNNs.

Dalle curve di training (Training accuracy e Validation accuracy) 4.1 si osserva che il modello raggiunge un'accuratezza molto elevata, del 99%, nell'identificazione delle 3 classi: SC, SA, PT. Inoltre, dalle curve rappresentanti le funzioni di perdita (Training Loss e Validation Loss) si osservano valori che decrementano fino a un buon 0.05%. Inoltre, è possibile notare dalle curve relative al training e alla validazione del modello (rispettivamente in blu e in arancio) che i risultati in fase di validazione del modello sono ottimali e senza presenza di overfitting del modello perché le curve mantengono valori prossimi l'un l'altra.

Dopo aver analizzato il modello che andremo ad utilizzare, evidenziamo di seguito la numerosità di casistiche correttamente classificate dallo stesso con un dataset di test. Questo è ottenuto estraendo il 20% dei dati iniziali sul volo 3, Zona A.
Nella precedente matrice di confusione 4.2, sulla diagonale che parte da sinistra a destra possiamo apprezzare il numero di profili correttamente determinati dal

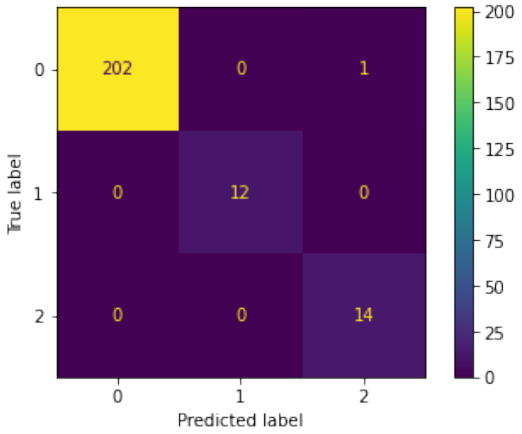


Figura 4.2: Matrice di confusione calcolata sul dataset di test del volo 3, Zona A

nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

I risultati trovati sono di 228 profili identificati correttamente contro 1 profilo errato su un totale di 229.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	1.00	1.00	1.00
label 1	1.00	1.00	1.00
label 2	0.93	1.00	0.97
accuracy			1.00
marco avg	0.98	1.00	0.99
Weighted avg	1.00	1.00	1.00

4.2 Esperimento 2

Il training è stato svolto sul volo 5, Zona B, con le metriche riportate sopra in Tabella 3.3. Si riportano dunque le curve del training del modello creato con 1D CNNs.

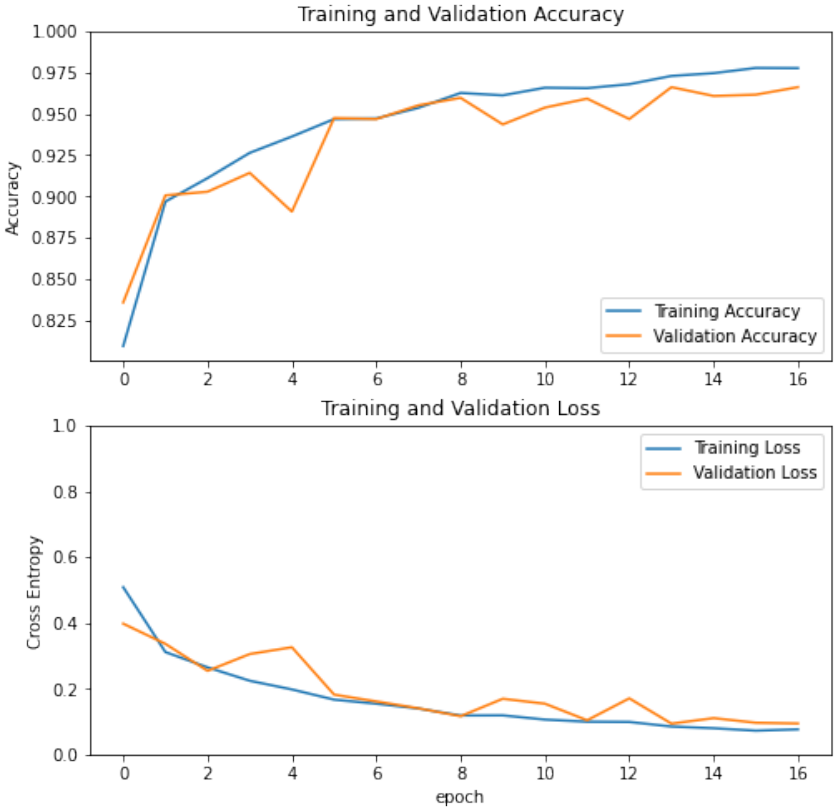


Figura 4.3: Curve di training e della funzione di perdita relative modello 5 allenato sul volo 5, Zona B

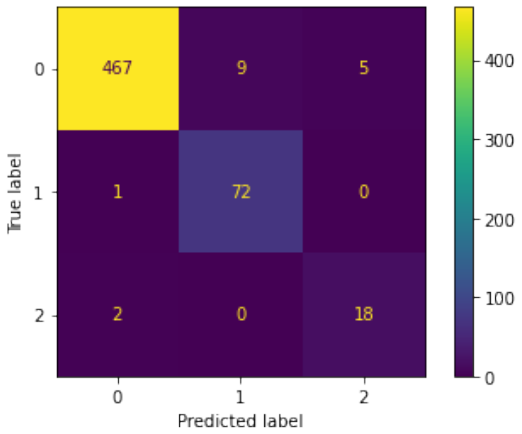


Figura 4.4: Matrice di confusione calcolata sul dataset di test del volo 5, Zona B

Dalle curve di training (Training accuracy e Validation accuracy) 4.3 si osserva che il modello raggiunge un'accuratezza molto elevata, del 96-97%, nell'identificazione delle 3 classi: SC, SA, PT. Inoltre, dalle curve rappresentanti le funzioni di perdita (Training Loss e Validation Loss) si osservano valori che decrementano fino a un buon 1.0%. Inoltre, è possibile notare dalle curve relative al training e alla validazione del modello (rispettivamente in blu e in arancio) che i risultati in fase di validazione del modello sono ottimali e con un overfitting trascurabile del modello perché le curve mantengono valori prossimi l'un l'altra.

Dopo aver analizzato il modello che andremo ad utilizzare, evidenziamo di seguito la numerosità di casistiche correttamente classificate dallo stesso con un dataset di test. Questo è ottenuto estraendo il 20% dei dati iniziali sul volo 5, Zona B.
Nella seguente matrice di confusione, sulla diagonale che parte da sinistra a destra possiamo apprezzare il numero di profili correttamente determinati dal nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

I risultati trovati sono di 557 profili identificati correttamente contro 17 profili errati su un totale di 574.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.99	0.98	0.98
label 1	0.89	0.99	0.94
label 2	0.78	0.90	0.84
accuracy			0.97
marco avg	0.89	0.95	0.92
Weighted avg	0.97	0.97	0.97

4.3 Esperimento 3

Il training è stato svolto sul volo 5, Zona B, con i parametri riportati sopra in Tabella 3.4. Si riportano dunque le curve del training del modello creato con 1D CNNs.

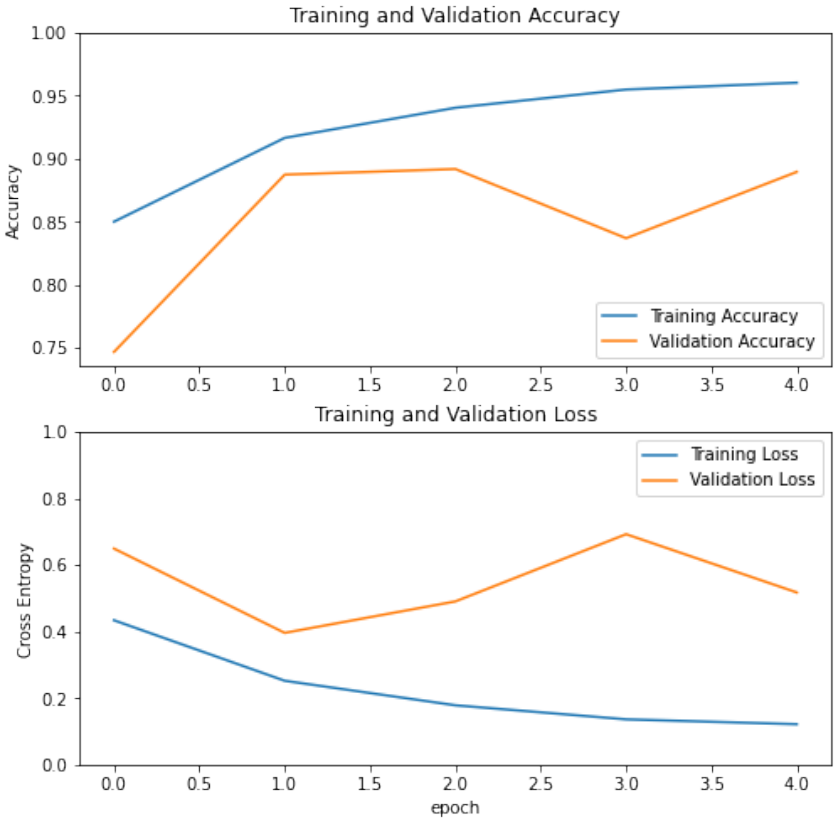


Figura 4.5: Curve di training e della funzione di perdita relative modello 5-8-7, allenato sul volo 5 (senza split del dataset), Zona B

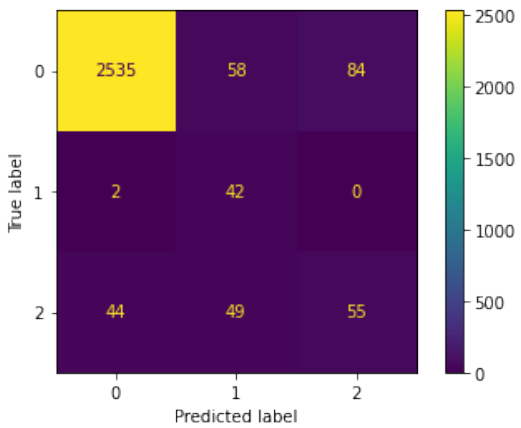


Figura 4.6: Matrice di confusione calcolata sul dataset del volo 8, Zona B

Dalle curve di training (Training accuracy e Validation accuracy) 4.5 si osserva che il modello raggiunge un'accuratezza molto elevata, del 96%, nell'identificazione delle 3 classi: SC, SA, PT. Inoltre, dalle curve rappresentanti le funzioni di perdita (Training Loss e Validation Loss) si osservano valori che decrementano fino a un buon 2.0%. Inoltre, è possibile notare dalle curve relative al training e alla validazione del modello (rispettivamente in blu e in arancio) che i risultati in fase di validazione del modello sono ottimali e con un overfitting del modello. La presenza di un overfitting maggiore rispetto ai casi precedenti è dovuta alla maggiore variabilità nel test su un volo differente da quello di training del modello.

Dopo aver analizzato il modello che andremo ad utilizzare, evidenziamo di seguito la numerosità di casistiche correttamente classificate dallo stesso con un dataset di test. Questo è ottenuto direttamente dal dataset sul volo 8, Zona B.
Nella seguente matrice di confusione, sulla diagonale che parte da sinistra a destra possiamo apprezzare il numero di profili correttamente determinati dal nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

I risultati trovati sono di 2632 profili identificati correttamente contro 237 profili errati su totale di 2869 profili.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.98	0.95	0.96
label 1	0.28	0.95	0.44
label 2	0.40	0.37	0.38
accuracy			0.92
marco avg	0.55	0.76	0.59
Weighted avg	0.94	0.92	0.93

4.4 Esperimento 4

Il training è stato svolto sul volo 5, Zona B, con i parametri riportati sopra in Tabella 3.5. Si riportano dunque le curve del training del modello creato con 1D CNNs.

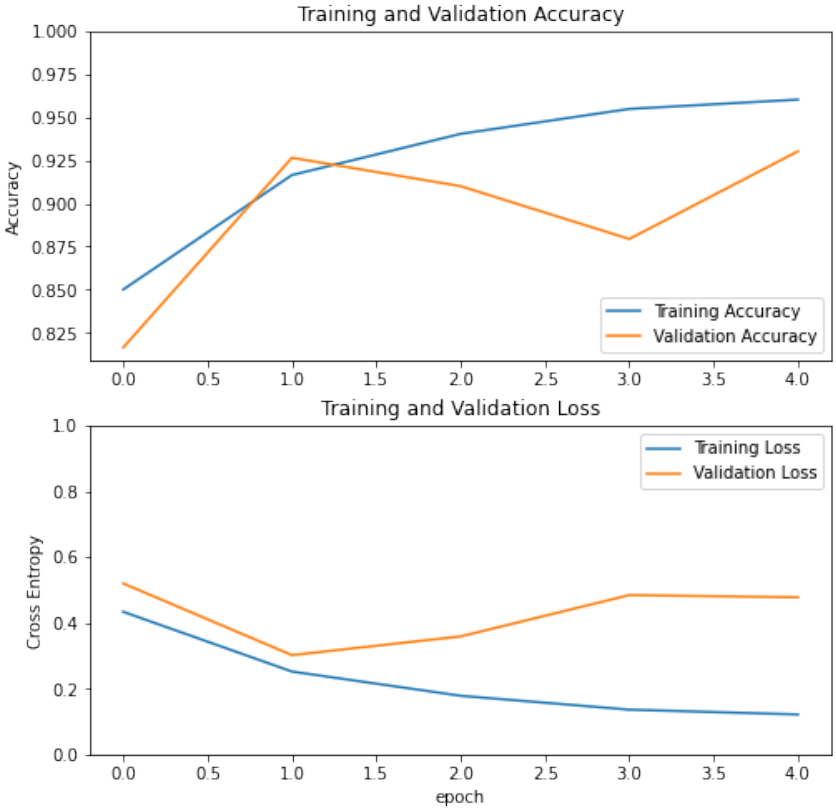


Figura 4.7: Curve di training e della funzione di perdita relative modello 5-7-8, allenato sul volo 5, Zona B

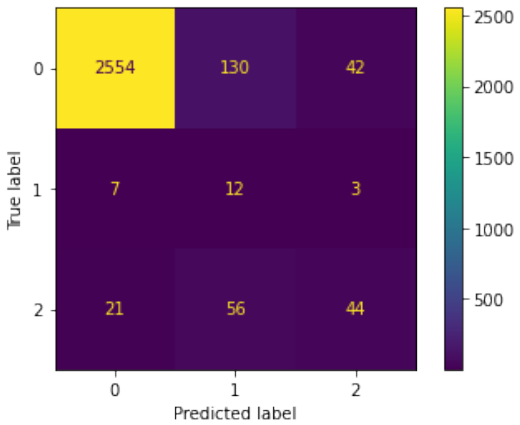


Figura 4.8: Matrice di confusione calcolata sul dataset del volo 7, Zona B

Dalle curve di training (Training accuracy e Validation accuracy) 4.7 si osserva che il modello raggiunge un'accuratezza molto elevata, del 96%, nell'identificazione delle 3 classi: SC, SA, PT. Inoltre, dalle curve rappresentanti le funzioni di perdita (Training Loss e Validation Loss) si osservano valori che decrementano fino a un buon 2.0%. Inoltre, è possibile notare dalle curve relative al training e alla validazione del modello (rispettivamente in blu e in arancio) che i risultati in fase di validazione del modello sono ottimali e con un overfitting del modello meno trascurabile rispetto ai casi precedenti perché le curve si discostano maggiormente. Ciò è dovuto alla variabilità maggiore nel test su un volo differente da quello di training del modello.

Dopo aver analizzato il modello che andremo ad utilizzare, evidenziamo di seguito la numerosità di casistiche correttamente classificate dallo stesso con un dataset di test. Questo è ottenuto direttamente dal dataset sul volo 7, Zona B. Nella seguente matrice di confusione, sulla diagonale che parte da sinistra a destra possiamo apprezzare il numero di profili correttamente determinati dal nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

I risultati trovati sono di 2610 profili identificati correttamente contro 259 profili errati su totale di 2869 profili.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.98	0.94	0.96
label 1	0.06	0.55	0.11
label 2	0.49	0.36	0.42
accuracy			0.91
marco avg	0.51	0.62	0.50
Weighted avg	0.96	0.91	0.93

4.5 Esperimento 5

Si riportano i risultati ottenuti da una generalizzazione effettuata sul volo 3 del cantiere in Zona A, utilizzando il modello 5 allenato sul volo 5 del cantiere in Zona B. In particolare, si mostra la matrice di confusione in cui sulla diagonale che parte da sinistra a destra si può apprezzare il numero di profili correttamente determinati dal nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

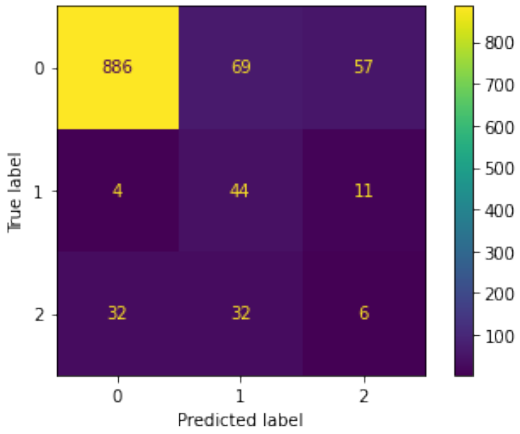


Figura 4.9: Matrice di confusione calcolata sul dataset del volo 3, Zona A, con il modello 5

I risultati trovati sono di 936 profili identificati correttamente contro 205 profili errati su totale di 1141 profili.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.99	0.79	0.88
label 1	0.22	0.78	0.34
label 2	0.15	0.26	0.19
accuracy			0.76
marco avg	0.45	0.61	0.47
Weighted avg	0.90	0.76	0.81

4.6 Esperimento 6

Si riportano i risultati ottenuti da una generalizzazione effettuata sul volo 3 del cantiere in Zona A, utilizzando il modello 5-8-7 allenato sul volo 5 del cantiere in Zona B.
In particolare, si mostra la matrice di confusione in cui sulla diagonale che parte da sinistra a destra si può apprezzare il numero di profili correttamente determinati dal nostro modello e, di fianco a questi, i profili scambiati per altri. In dettaglio, sull'asse orizzontale è indicata la numerosità reale dei profili in relazione alla classe di appartenenza, mentre sull'asse verticale la numerosità predetta dei profili.

I risultati trovati sono di 865 profili identificati correttamente contro 276 profili errati su totale di 1141 profili.

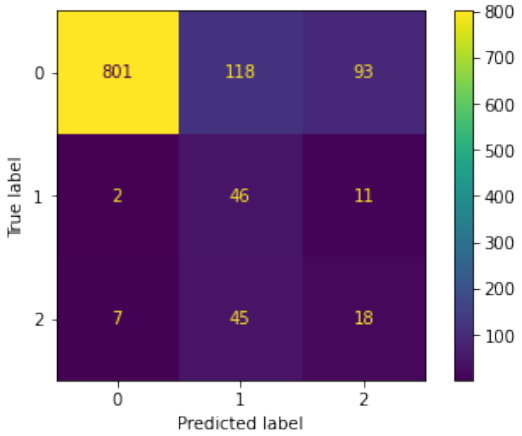


Figura 4.10: Matrice di confusione calcolata sul dataset del volo 3, Zona A, con il modello 5-8-7

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.96	0.88	0.92
label 1	0.30	0.75	0.43
label 2	0.08	0.09	0.08
accuracy			0.82
marco avg	0.45	0.57	0.48
Weighted avg	0.87	0.82	0.84

4.7 Esperimento 7

Si ricorda che in questo esperimento è stato ampliato il dataset iniziale di training per il modello 5-8. Ciò è avvenuto sommando i dati dei voli 5 e 8 sul cantiere in Zona B. Invece, si utilizza il volo 7 per validarlo. Di seguito, le curve di training del modello 5-8:

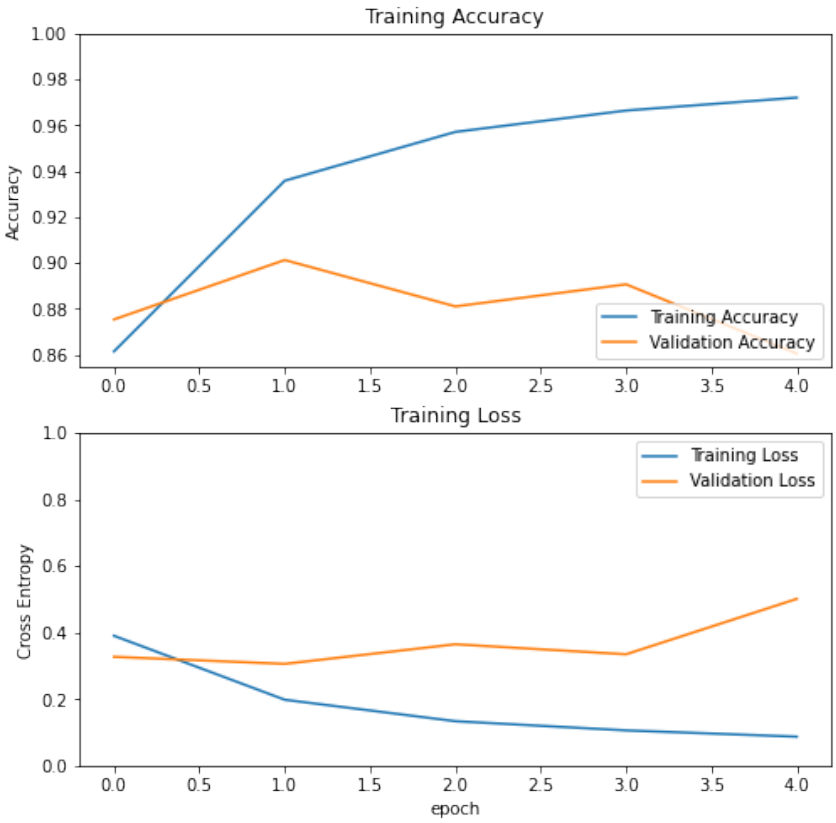


Figura 4.11: Curve di training del modello 5-8

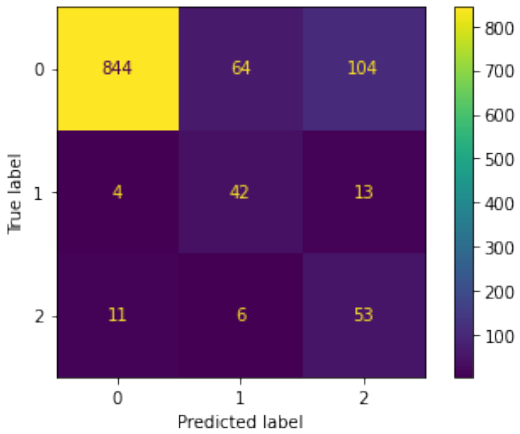


Figura 4.12: Matrice di confusione calcolata sul dataset del volo 3, Zona A, con il modello 5-8

I risultati trovati sono di 939 profili identificati correttamente contro 202 profili errati su totale di 1141 profili. Dunque, con il modello 5-8 si è ottenuto un miglioramento nell'identificare correttamente le tre classi. Questo rende evidente che aumentando i dati con cui il modello viene allenato, i risultati che questo produrrà saranno migliori.

Inoltre, si riporta per completezza il report sulla classificazione effettuata:

	precision	*recall*	*f1-score*
label 0	0.98	0.83	0.90
label 1	0.38	0.71	0.49
label 2	0.31	0.76	0.44
accuracy			0.82
marco avg	0.56	0.77	0.61
Weighted avg	0.91	0.82	0.85

Anche i risultati percentuali di precisione nell'identificazione delle singole classi vedono un aumento considerevole rispetto agli esperimenti precedenti 5 (4.5) e 6 (4.6).

4.8 Visualizzazione risultati

Per mostrare i risultati ottenuti tramite il software QGIS, si utilizzano due operazioni che il software rende accessibili dalle proprietà di ciascun layer:

- *join* di un layer con una sorgente dati. Si tratta di effettuare la join del layer «output-lines», contenente tutti i profili, e del file .txt ottenuto come output dell'algoritmo di DL. L'unione avviene tramite il parametro ID, fig.4.13.

- *simboleggiatura* dei dati ricavati dalla join, in particolare dal parametro LABEL all'interno del file .txt *elevation-outdata*, fig. 4.14.

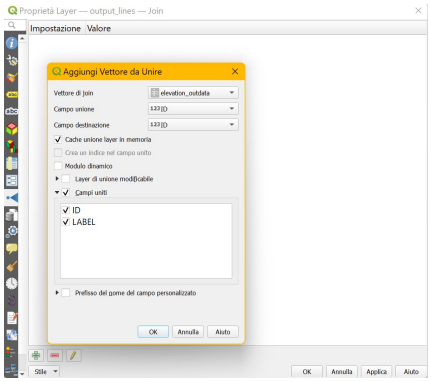


Figura 4.13: Operazione di *join* in QGIS

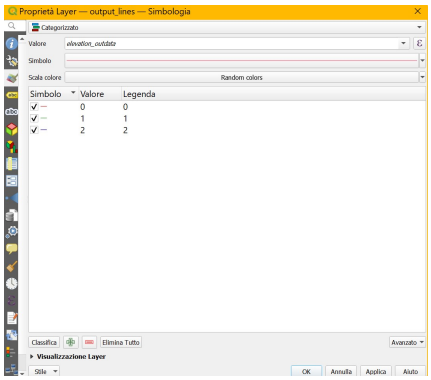


Figura 4.14: Operazione di *simboleggiatura* in QGIS

Si mostrano dunque le ortofoto (fotografie aeree georeferenziate) dei due cantieri analizzati con in rilievo i profili a colori. Ciascun colore indica uno stato differente predetto dalla rete neurale:

- rosso per SC;
- verde per SA;
- blu per PT.

Di seguito, due esempi di corrette classificazioni da parte del modello sul cantiere B, in fig. 4.15, 4.16.

Di contro, invece, si riportano due esempi di classificazioni erronee: in fig. 4.17 il modello scambia un intervallo di scavo chiuso in cui è presente un mezzo pesante per uno scavo aperto (linee evidenziate in verde). Mentre, in fig. 4.18 il modello scambia dei profili con il tubo in posa per uno scavo aperto a causa di una bassa risoluzione dell'immagine originaria.

Figura 4.15: Esempio di corretta classificazione di scavo aperto

Figura 4.16: Esempio di corretta classificazione di posa del tubo

Figura 4.17: Errata classificazione causata dalla presenza di un mezzo pesante

Figura 4.18: Errata classificazione causata dalla bassa risoluzione dell'ortofoto originaria

4.8.1 Confronto tra i risultati finali sul cantiere A. Risultati visuali ottenuti in relazione al volo 3 del cantiere A.

In figura 4.19 si mostra visivamente la classificazione effettuata dal modello 5, creato interamente a partire dal volo 5 del cantiere B. I risultati sono infatti coerenti con quelli discussi nell'Esperimento 3.7.5.

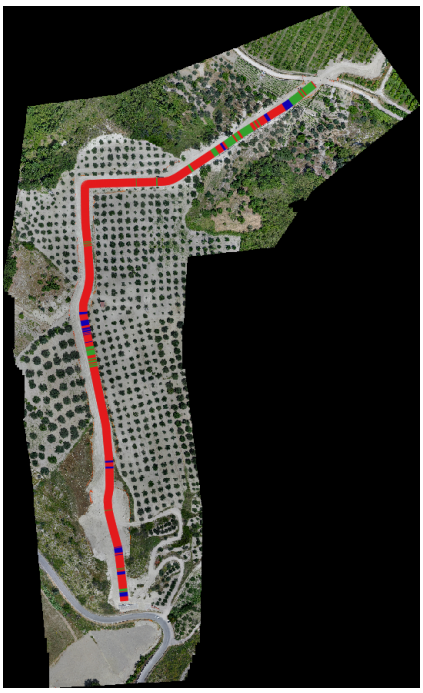


Figura 4.19: Ortofoto del volo3, Zona A, predizioni del modello 5

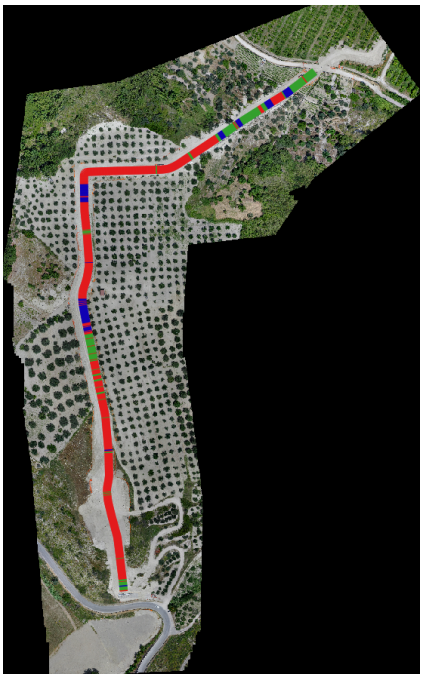


Figura 4.20: Ortofoto del volo3, Zona A, predizioni del modello 5-8-7

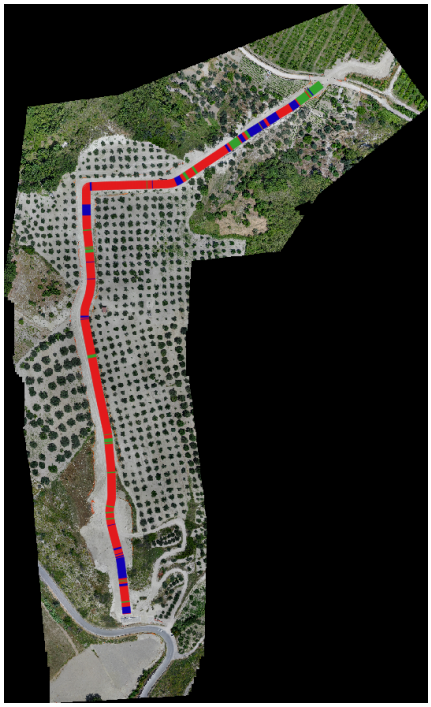


Figura 4.21: Ortofoto del volo3, Zona A, predizioni del modello 5-8

Figura 4.22: Dettaglio ortofoto del volo 3, Zona A, predizioni del modello 5-8

Figura 4.23: Confronto con ortofoto del volo 3, Zona A, predizioni del modello 5

Invece, in figura 4.20, si mostra visivamente la classificazione effettuata dal modello 5-8-7, creato utilizzando i voli 5 e 8 sul cantiere B. I risultati sono infatti coerenti con quelli espressi nell'Esperimento 3.7.6.

Infine, in figura 4.21, si mostra visivamente la classificazione effettuata dal modello 5-8, creato unendo i dati dei voli 5 e 8 sul cantiere B. I risultati sono i migliori tra i tre modelli discussi, come ben evidente nell'Esperimento 3.7.7. Infatti nel dettaglio sul modello 5-8, in figura 4.22, è evidenziata la corretta classificazione del tubo in posa, a meno di piccoli sbagli trascurabili. Ciò è evidente guardando il dettaglio sulla predizione del modello 5 in figura 4.23; questa confonde in maggior parte l'intervallo di PT per SA (in verde).

4.8.2 Confronto tra i risultati iniziali e i risultati finali sul cantiere B

Risultati visuali ottenuti in relazione a due differenti volo sul cantiere B: volo 7, volo 8.

Ortofoto Volo 7

In figura 4.24 si mostra visivamente la classificazione effettuata dal modello 5-7-8, creato a partire dai voli 5 e 7 del cantiere B. I risultati sono infatti coerenti con quelli discussi nell'Esperimento 3.7.3.

Ortofoto Volo 8

In figura 4.25 si mostra visivamente la classificazione effettuata dal modello 5-8-7, creato a partire dai voli 5 e 8 del cantiere B. I risultati sono infatti coerenti con quelli discussi nell'Esperimento 3.7.4.

Figura 4.24: Ortofoto del volo 8, Zona B, predizioni del modello 5-7-8

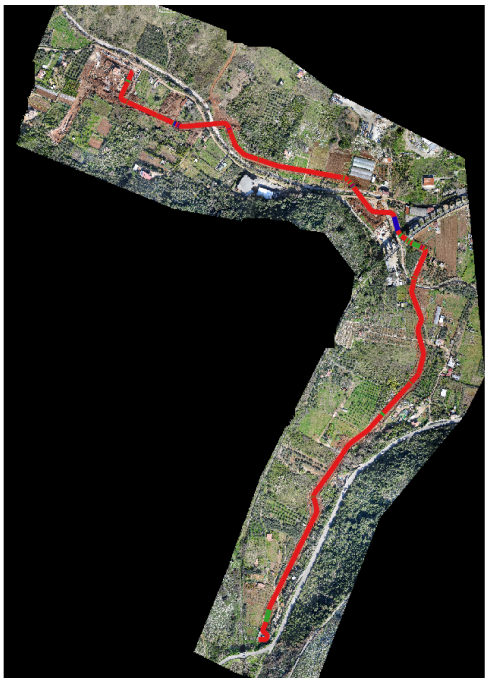


Figura 4.25: Ortofoto del volo 7, Zona B, predizioni del modello 5-8-7

Capitolo 5

Discussioni

Dai risultati ottenuti si nota che con l'aumentare la dimensione del dataset di training, e in particolare la sua variabilità, si ottiene un miglioramento nella classificazione su un cantiere differente. Questo è prevedibile considerando che il modello viene allenato ad identificare profili con una natura più varia, e perciò risulta maggiormente efficace. A influire sui risultati vi è certamente la presenza di fattori che differenziano un cantiere da un altro; questi non possono essere trascurati nelle valutazione che riguarda la generalizzazione di un particolare modello su di un altro cantiere.

I risultati ottenuti sono quindi positivi e lasciano spazio a miglioramenti. Infatti, si ha una buona accuratezza delle predizioni, specialmente per le classi maggioritarie nelle tre generalizzazioni considerate: su uno stesso volo, su un altro volo, su un altro cantiere.

In particolare, si evidenziano due aspetti importanti che rendono i tre casi diversi tra loro:

- Quando si estrae il test dataset direttamente da uno stesso volo su un determinato cantiere, la precisione nell'identificare le classi SC, SA e PT si aggira fra il 97% e il 100%. Questo risultato ottimale è in parte dovuto al fatto che, restando su uno stesso volo, i dati analizzati in ingresso (DSM e shapefile) sono ben omogenei.

- D'altro canto, i dati in ingresso differiscono maggiormente quando il test dataset è un volo diverso da quello utilizzato per allenare il modello. Ciò produce errori più o meno grandi in fase di classificazione con un'accuratezza che in media si aggira al 91%, con percentuali nettamente più basse per le classi minoritari, che sono SC e PT.

Nell'ultima tipologia di generalizzazione su un altro cantiere, ci si pone in un caso di utilizzo reale del modello. Questo viene allenato su un particolare cantiere, e conseguentemente impiegato per fare predizioni sul volo di un nuovo cantiere. Si ottengono buone prestazioni dei modelli anche in questi casi, con un notevole incremento percentuale di classificazione corretta delle classi minoritarie (SC e PT) nell'Esperimento 3.7.7 rispetto ai precedenti Esperimenti 3.7.5 e 3.7.6. Infatti, nell'ultimo esperimento realizzato i dati in input sono il doppio rispetto ai casi precedenti

perché ricavati dall'unione di due diversi voli su uno stesso cantiere. La conseguenza è che il modello è stato allenato a riconoscere un maggior numero di casistiche differenti, migliorandone visibilmente le capacità.

In questa sperimentazione, si è passati da una classificazione binaria ad una multi-classe nell'ambito dell'identificazione dello stato avanzamento dei lavori di un cantiere. Questo iter illustrato nei capitoli precedenti, lascia spazio alla definizione di nuove caratteristiche identificative e quindi ad una maggior capacità della rete neurale di stabilire lo stato avanzamento dei lavori.

Capitolo 6

Conclusioni

L'obiettivo definito all'inizio dello studio è stato completato con successo grazie alla realizzazione di un algoritmo di DL che svolge una classificazione multi-classe dei profili di scavo chiuso, aperto e di posa del tubo. Di seguito, si rimarcano i passi chiave dell'attuazione del progetto.

Il punto di partenza sono le ortofoto dei cantieri (fotografie geometricamente corrette) e i DSM dei voli sui cantieri (immagini digitali del terreno). A seguire, si procede con l'analisi dei dati che consiste nella creazione di nuovi dati importanti per lo sviluppo di modelli della rete neurale; tra questi sicuramente gli shapefiles e i profili dei DSM. I primi non sono altro che polilinee che percorrono il tracciato dei lavori mentre i secondi sono i segmenti orizzontali alle polilinee. Processando quindi questi dati si arriva a generare un file di tipo JSON che contiene i profili relazionati alle classi di appartenenza. Questo file (*"ElevationValues-full.json"*) contiene le coordinate di elevazione di ogni profilo e un numero che ne indica l'appartenenza ad una determinata classe e sarà l'input per il training dei modelli delle ANN.

A questo punto si prosegue con la creazione di CNN. Infatti i modelli che effettueranno predizioni su nuovi set di dati mai visti prima dal modello sono elaborati tramite layers convoluzioni e fully-connected insieme a tecniche di data augmentation, bilanciamento delle proporzioni tra le classi tramite un'apposita funzione, oltre ad appositi split del dataset.

Infine, si integrano i file TXT in output con il software QGIS in modo tale da poter visualizzare i vari profili distinti per ID e con un colore diverso a seconda della predizione numerica della classe di appartenenza.

I risultati sono stati raccolti su un totale di 7 esperimenti:

- Esperimenti 1 e 2 che riguardano generalizzazioni su uno stesso volo.
- Esperimenti 3 e 4 che riguardano generalizzazioni su uno altro volo.
- Esperimenti 5, 6 e 7 che riguardano generalizzazioni su un altro cantiere.

Attraverso questi esperimenti si è evidenziato che nel primo caso i risultati sono ottimali, in quanto l'accuratezza di predizione dei modelli è molto vicina a 1 senza overfitting dei modelli.

Nella seconda casistica di test dei modelli su dataset derivanti da atri voli, come prevedibile per le differenze sostanziali che si riscontrano tra due voli, si ha una diminuzione nell'accuratezza delle predizioni, ma questa rimane comunque tendente ad 1, sebbene si presenti un leggero overfitting dei modelli.

Invece, nella terza ed ultima casistica si sperimentano modelli creati su un cantiere (cantiere Zona B) diverso da quello di test (cantiere Zona A). In particolare, i dati selezionati per il training, la validation e il test dei modelli sono ricavati da tre diversi voli su un cantiere. L'intenzione è infatti quella di creare un esperimento più vicino al caso reale, in cui il modello viene testato sul dataset di un volo effettuato su un cantiere sconosciuto al modello. I risultati di questi esperimenti presentano un'accuratezza inferiore a 0.5 per le classi minoritarie che sono quelle di SC e PT. Chiaramente la nuova difficoltà riscontrata per le predizioni dei modelli di ANN è il test su un volo relativo ad un cantiere completamente differente da quello di allenamento del modello.
Inoltre si osservi che l'ultimo esperimento è differente dai precedenti due perché utilizza due voli (5,8 Zona B) come dataset di training e un volo come dataset di validation (7 Zona B). Ciò ha portato ad un miglioramento consistente delle percentuali di accuratezza di classificazione rispetto ai due casi precedenti. Infatti questi risultati si accostano maggiormente a quelli desiderati in quanto l'accuratezza di classificazione aumenta e lo squilibrio tra le classi minoritarie è trascurabile. Ciò è attribuibile al fatto che il modello ha "visto" il doppio delle casistiche in fase di training rispetto i precedenti esperimenti e quindi ha acquisito una maggiorare capacità di classificare correttamente.

In pratica, al termine dello studio, si hanno tre modelli di ANN già allenati e creati partendo da due cantieri. Si evidenziano i casi di "model3" il cui dataset è prelevato da un singolo volo e diviso tra training, test e validation (come specificato in 3.7.1); "model5-7-8" il cui dataset è ricavato da più voli su uno stesso cantiere (come specificato in 3.7.4); e infine "model5-8" (paragrafo 3.7.7) che, analogamente al caso precedente, ricava il dataset a partire da più voli su uno stesso cantiere, ma questa volta tutti e tre i voli sono riservati per la parte di training e di validation mentre il test viene effettuato sul volo di un cantiere differente.

6.1 Vantaggi e Limitazioni

Questo approccio adottato ha permesso di valutare risultati di interesse come quelli relativi ad un nuovo volo effettuato su uno stesso cantiere oltre ad un nuovo volo relativo ad un altro cantiere. Così facendo, si è ricercato progressivamente di ottenere risultati desiderabili attraverso le casistiche analizzate e utilizzando procedimenti replicabili anche per un numero maggiore di classi.

In aggiunta l'utilizzo di reti 1D CNNs ha comportato una complessità computazionale lineare pari a O(KN) rispetto alle alternative 2D CNNs, che presentano una complessità quadratica $O(K^2N^2)$. Questo significa che a parità di determinate

condizioni quali configurazione, rete e iper-parametri, la complessità computazionale di una 1D CNN è di gran lunga inferiore ad una 2D CNN [23]. Inoltre, in genere il training di 2D CNN richiede attrezzature hardware particolari mentre per configurazioni di 1D CNN compatte con un numero ridotto di hidden layers (e.g. 1 o 2) e neuroni (e.g. 50) è possibile fare il training di un modello con una qualsiasi CPU di un computer moderno in maniera relativamente veloce. Un ulteriore vantaggio riguarda come questa tipologia di CNN sia adatta per applicazioni real-time, specialmente se si tratta di dispositivi mobili o hand-hend, quindi adatta a generare predizioni in tempo reale su un cantiere, limitatamente alle capacità di elaborazione del dispositivo.

Alcune limitazioni esterne hanno ridotto l'accuratezza delle classificazioni prodotte dalle ANN; due sono gli aspetti che hanno influito maggiormente e che riguardano i dati di input:

- rumorosità delle acquisizioni; queste sono dovute alla presenza di mezzi pesanti o costruzioni lungo il tracciato del cantiere. Chiaramente, informazioni di questo genere portano ad errate interpretazioni di classificazione dell'algoritmo poiché gli oggetti presenti non risultano classificabili tra le classi definite.
- eterogeneità dei voli; infatti, dalle ortofoto è evidente come i voli sono stati effettuati in diverse ore del giorno poiché presentano zone d'ombra differenti, più o meno accentuate. Questo comporta che alcune zone del tracciato rimangono oscurate e siano di difficile interpretazione per l'algoritmo.

6.2 Futuri sviluppi

In quest'ultimo paragrafo, si parla dei possibili sviluppi in corso o futuri che sono realizzabili facendo uso del lavoro svolto in questa dissertazione.

- Come viene suggerito dai risultati ottenuti, una possibilità è quella di raccogliere e fornire un numero considerevolmente maggiore di dati per il training del modello, derivanti da vari voli su un cantiere al fine di aumentarne considerevolmente le capacità di identificare correttamente le classi definite.
- Così come la semplice espansione ad un numero maggiore di classi che comprendano nuovi stati riguardanti i lavori all'interno di un cantiere, come ad esempio la classificazione di un tubo posato a fianco dello scavo e pronto per essere interrato.
- È già in fase di realizzazione l'integrazione del sistema di DL creato con un framework che includa e automatizzi il processo nel suo complesso. L'obiettivo ultimo è la creazione di un contenitore che funga da black-box, il quale ricevendo in input i dati raccolti dai droni relativamente ad un nuovo volo su un cantiere restituisca in output la classificazione effettuata sullo stato dei lavori in corso.

- Si aggiunge inoltre un'osservazione volta al miglioramento della raccolta dati tramite droni: mezzi pesanti lungo il tracciato dei lavori o zone offuscate da condizioni meteorologiche inadatte alla cattura di immagini posso precludere un riconoscimento ottimale da parte delle ANN; perciò si suggerisce di prestare attenzione a questo dettaglio durante la fase raccolta dati.

Bibliografia

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[6] Marco Gori. "Introduzione alle reti neurali artificiali". In: (2003).

[7] Christian Janiesch, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning". In: *Electronic Markets* 31.3 (2021), pp. 685–695.

[8] Ashley Varghese et al. "ChangeNet: A Deep Learning Architecture for Visual Change Detection". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018. URL: https://link.springer.com/chapter/10.1007/978-3-030-11012-3_10.

[9] Dongyeob Han et al. "Change Detection in Unmanned Aerial Vehicle Images for Progress Monitoring of Road Construction". In: *Buildings* 11.4 (2021). ISSN: 2075-5309. DOI: 10.3390/buildings11040150. URL: https://www.mdpi.com/2075-5309/11/4/150.

[10] Roberto Minervini. "Progettazione e Sviluppo di un algoritmo per la valutazione automatica dello stato di avanzamento di un cantiere mediante immagini da drone". In: (2022).

[20] GNU General Public License. "Gnu general public license". In: *Retrieved December* 25 (1989), p. 2014.

[23] Serkan Kiranyaz et al. "1D convolutional neural networks and applications: A survey". In: *Mechanical systems and signal processing* 151 (2021), p. 107398.

[25] Hidenori Ide and Takio Kurita. "Improvement of learning for CNN with ReLU activation by sparse regularization". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 2684–2691.

[26] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[27] Diederik P Kingma. "&Ba J.(2014). Adam: A method for stochastic optimization". In: (2015).

[30] Guy Lapalme Marina Sokolova. "A systematic analysis of performance measures for classification tasks". In: *Information Processing and Management* (2009), p. 430.

Sitografia

[1] Drone senseFly eBee. https://ageagle.com/drones/ebee-x/.

[2] DJI Mini 2. https://www.dji.com/it/mini-2/specs.

[3] ENAC - Ente Nazionale per l'Aviazione. https://www.enac.gov.it/sicurezza-aerea/droni.

[4] Litchi. https://flylitchi.com/.

[11] Google Colab. https://colab.research.google.com/.

[12] Keras. https://keras.io/.

[13] Scikit-learn. https://scikit-learn.org/stable/.

[14] QGIS. *A Free and Open Source Geographic Information System*. https://www.qgis.org/en/site/.

[15] MAPASYST. *What is an orthophoto?* https://mapasyst.extension.org/what-is-an-orthophoto/. 2019.

[16] GISGeography. *DEM, DSM DTM Differences – A Look at Elevation Models in GIS*. https://gisgeography.com/dem-dsm-dtm-differences/.

[17] JSON. https://www.json.org/json-en.html.

[18] GIS Resources. *Understanding Shapefile (.shp) File Format*. https://gisresources.com/understanding-shapefile-shp-file-format/. 2014.

[19] ESRI. *ESRI Shapefile Technical Description*. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf. 1998.

[21] Google Drive. https://www.google.com/drive/.

[22] Python. *Python 3.8.10 release*. https://www.python.org/downloads/release/python-3810/.

[24] Keras. *Conv1D layer*. https://keras.io/api/layers/convolution_layers/convolution1d/.

[28] Keras. *EarlyStopping*. https://keras.io/api/callbacks/early_stopping/.

[29] Keras. *ModelCheckpoint*. https://keras.io/api/callbacks/model_checkpoint/.