

GeoAI Report group 5

Riccardo Fragale
s307295

Maurizio Lanzoni
s313173

Lucia Perri
s286547

Giulio Bosco
s334925

Abstract—Tra i molteplici campi di applicazione dell'intelligenza artificiale vi è quello dei dati geospaziali. In particolare, è possibile allenare una rete neurale per riconoscere alcuni pattern contenuti in immagini satellitari e in rappresentazioni tridimensionali dello spazio quali ad esempio le nuvole di punti. All'interno di questo report analizzeremo alcune possibili applicazioni di questa nuova frontiera della geomática, concentrandoci su alcuni casi studio visti in aula durante le esercitazioni e su un paio di approfondimenti. La finalità principale è valutare le performance dei modelli di intelligenza artificiale applicati ai dati geospaziali, incluse possibili migliorie da applicare per migliorare i risultati ottenuti e rendere questi strumenti ancor più efficaci.

I. GEOMATICA 2D

A. Descrizione del task

Il primo lavoro prevedeva la classificazione di un'immagine satellitare. Questa tipologia di task consiste nel predire, ricavato un dato in input, la classe a cui esso appartiene. In questo caso, essendo il dato in ingresso un raster, è stata attribuita una classe ad ogni singolo pixel. In ambito GeoAI, gli algoritmi capaci di svolgere questo tipo di task risultano essere utili per monitorare i cambiamenti del territorio, sia a livello locale sia a livello globale. La classificazione permette ad esempio di fare mapping delle tipologie di territorio, i cui risultati sono fondamentali ai fini della pianificazione urbanistica (smart cities), oppure, in ambito climatico-ambientale, consente di mappare i ghiacciai o le fonti rinnovabili, riuscendo ad individuare inoltre eventuali zone danneggiate a seguito di eventi catastrofici. A tal proposito, a seguito dell'esercitazione condotta in aula, è stato svolto un approfondimento che ha avuto come obiettivo il confronto tra due raster classificati, rappresentanti la città di Valencia rispettivamente prima e dopo l'alluvione avvenuta il 29 ottobre 2024.

B. Esercitazione svolta in aula

Partendo dall'esercitazione, al fine di individuare all'interno dell'immagine le quattro classi richieste, ovvero acqua, vegetazione, costruiti e terreni, è stato utilizzato il software QGIS.



Fig. 1: Legenda classi

C. Descrizione del dataset

I classificatori testati sono stati allenati utilizzando lo stesso dataset, creato manualmente attraverso la definizione di un gruppo di ROI, ‘Regions Of Interest’, selezionando aree dell'immagine satellitare rinvenuta da Sentinel 2A, alle quali veniva da noi associata la classe di appartenenza: guardando la figura sottostante, esse corrispondono alle aree rosse. Per ciascuna classe sono quindi state individuate le regioni contenenti le features più significative, grazie alle quali i modelli potevano imparare a discriminare le classi al meglio.



Fig. 2: ROI di test (rosso) e validation (azzurro)

L'immagine presa dal satellite è stata caricata sul software, andando a combinare le bande B1, B2, B3, B4, B5, B6, B7, B8, B8A, B9, B11 e B12. L'immagine è georeferenziata con il sistema di coordinate WGS84. La risoluzione è di circa 15.5 m per pixel.

D. Modelli utilizzati ed analisi delle Performance del Modello

Sono stati testati tre diversi modelli al fine di confrontare le predizioni di classificazione e scegliere di conseguenza il modello più efficiente, dando non solo importanza all'accuratezza, bensì anche ai tempi di lavoro, essendo essi sicuramente correlabili al carico computazionale. Dapprima è stato utilizzato il Random Forest (RF), un modello che al suo interno racchiude a sua volta modelli più semplici, alberi decisionali, allenati parallelamente attraverso insiemi di dati ottenuti da quelli iniziali tramite campionamento casuale con rimpiazzo o bagging. Esso è governato quindi dalla logica di ensemble learning, per la quale, la predizione finale è ottenuta riunendo i risultati dei vari alberi decisionali. In questo modo si vuole disincentivare l'overfitting, ed ottenere quindi una rete capace di generalizzare e di essere poco sensibile agli eventuali cambiamenti che si possono avere all'interno di un dataset, conseguentemente l'aggiunta di outliers ad esempio. Sono stati fatti 4 test, in cui il ‘number of trees’, unico parametro che veniva modificato, era pari a 150, 200, 300 e 400; i risultati delle differenti configurazioni non hanno presentato discrepanze rilevanti dal punto di vista qualitativo, mentre per quanto riguarda le tempistiche, il RF con 400 alberi ha

impiegato per la predizione 15 minuti in più rispetto al RF con 150 alberi.



Fig. 3: Comparazione dei risultati

Durante la fase di creazione del dataset di training, sono state create nuove ROI fin tanto che la predizione finale del RF non fosse abbastanza soddisfacente. In particolare, sono state realizzate molte ROI con lo scopo di insegnare al modello a discriminare la classe dell'acqua dalla classe del costruito, spesso confuse nei pixel soggetti a riflesso, probabilmente perché la rete individuava in loro features simili, come ad esempio il colore bianco lucente. Per la classe dei terreni sono state selezionate ROI più grandi, in modo tale da individuare pattern più ampi, come ad esempio la loro caratteristica quadrettatura colorata.



Fig. 4: Difficoltà di riconoscimento dei fiumi classificati come costruito



Fig. 5: Campi classificati erroneamente come vegetazione

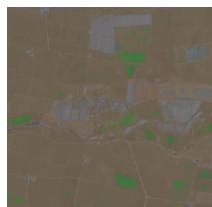


Fig. 6: Soluzione classificazione errata dei campi: creazione ROI più ampie

Successivamente è stata utilizzata la rete neurale artificiale chiamata Multi Layer Perceptron (MLP), un insieme di layer di neuroni addestrati seguendo la logica dello stochastic gradient descent. Brevemente, la rete in questione viene allenata fino a quando non viene trovata una sua configurazione per la quale la funzione costo è minimizzata, e conseguentemente gli errori commessi sono di numero limitato. In questo caso, il modello è stato usato tre volte, modificando solo il valore del learning rate, uno degli iperparametri più importanti, rappresentante il passo con il quale i pesi della rete vengono aggiornati durante il processo di minimizzazione del gradiente. Per quanto riguarda le tempistiche, il MLP è risultato in generale più veloce rispetto al RF. In termini di classificazione, si è notato un miglioramento nel riconoscimento delle classi acqua e costruito con il diminuire del valore del learning rate, partendo da 0.1, questo congruentemente con il fatto che un basso valore del learning rate in generale rallenta il processo di aggiornamento dei pesi della rete ma al contempo affina le sue capacità di apprendimento; il parametro non deve essere però troppo piccolo per non rischiare che durante il gradient descent si rimanga bloccati in un minimo locale della funzione di errore. Inoltre, la configurazione con il learning rate minore è riuscita a riconoscere molto bene le strade.



Fig. 7: Comparazione dei risultati per le classi costruito e acqua

Infine è stato utilizzato un terzo modello, ovvero il Support Vector Machine (SVM), un altro tipo di classificatore il quale separa idealmente i dati di addestramento nelle diverse classi di appartenenza attraverso la creazione di un iperpiano; il livello di tolleranza del margine di separazione dei dati è governato dal parametro C, per il quale si è scelto un valore unitario, in modo tale da poter avere un'alta varianza e tollerare eventuali classificazioni errate. Il SVM ha fornito la predizione dopo circa 60 minuti. A livello qualitativo è il modello che meglio ha riconosciuto le strade, notando però un mal funzionamento nella differenziazione tra le classi vegetazione e terreno.



Fig. 8: Comparazione dei risultati per la classe strade

Di seguito viene fornita una tabella riassuntiva, nella quale vengono confrontate le accuracy ed i tempi di lavoro dei modelli utilizzati. Il livello di accuratezza è stato calcolato andando a testare i modelli su ROI di validation, ovvero aree dell'immagine satellitare ancora non osservate dalle reti (vedi aree azzurre nell'immagine sopra).

TABLE I: Confronto vari modelli

Modelli	Accuracy	Time(min)
RF 150trees	93,8950%	10
RF 200trees	94,5777%	12,5
RF 300trees	94,5936%	19,5
RF 400trees	94,5839%	25,5
MLP lr0.0001	96,2392%	4,5
MLP lr0.0010	95,7551%	2,5
MLP lr0.1000	90,4652%	1
SVM	85,3772%	60

Come si può notare il livello di accuracy minore è risultato essere quello del SVM a causa, come detto precedentemente, dell'errata classificazione di porzioni di terreno come vegetazione; ciò si riscontra nella matrice di errore sottostante.

SVM	Reference			
	Water	Vegetation	Built-up	Soil
Classified				
Water	2268	0	53	0
Vegetation	3	3312	0	4814
Built-up	62	0	2218	26
Soil	2	20	748	13474

Fig. 9: Matrice di errore SVM

Si riportano sotto le predizioni della configurazione migliore per ciascun modello con a lato l'immagine satellitare.

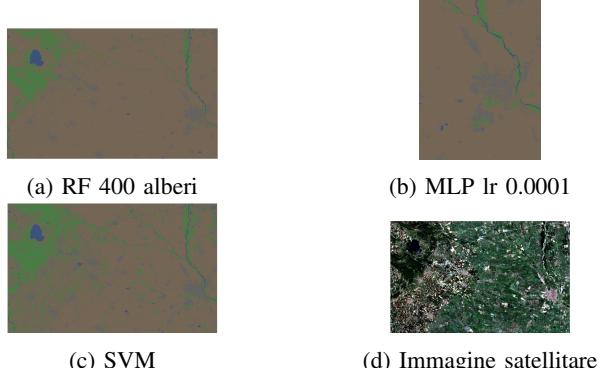


Fig. 10: Confronto predizioni

II. APPROFONDIMENTO VALENCIA

Al fine di poter comparare in maniera più intuitiva le situazioni pre e post alluvione di Valencia avvenuta il 29 ottobre 2024, si è deciso di classificare l'immagine differenziando quattro classi: acqua, vegetazione, costruito e terreno: quest'ultimo comprendeva campi, spiagge, zone costiere e montagne. Dal sito di Copernicus sono state scaricate le immagini rilevate dal satellite Sentinel2A rispettivamente nelle

date del 26 ottobre 2024 e del 10 novembre 2024, le quali sono risultate le migliori ricavabili in termini di copertura nuvolosa (0%). Ad esempio, non è stata scelta quella del 30 ottobre, in quanto la copertura nuvolosa avrebbe influito negativamente sulla classificazione. Sono state utilizzate le stesse bande dell'esercitazione precedentemente descritta e la risoluzione delle immagini è di circa 14 metri per pixel. Nonostante l'utilizzo dei tre classificatori RF, MLP e SVM, sotto viene riportata solo la predizione del RF settato con number of trees pari a 100 (RF 100); questo perché le ROI utilizzate per il training dei modelli sono state scelte in modo tale da ottimizzare il funzionamento di quest'ultimo. Testando le altre due reti sullo stesso dataset, si è rilevata una cattiva classificazione in particolare in prossimità del confine mare-spiaggia. Inoltre il modello di SVM ha fornito la sua predizione solo dopo un'ora di lavoro. Da notare inoltre che la classificazione di questi raster ha richiesto in generale tempi maggiori rispetto a quelli impiegati durante l'esercitazione, probabilmente per la presenza di molte zone piccole caratterizzate dalla compresenza delle 4 classi: RF 100 ha impiegato circa 13 minuti. Le ROI di training sono state create con l'ulteriore obiettivo di far imparare al modello RF 100 a riconoscere le strade, in modo tale da poter capire quali di esse sono state colpite dall'alluvione.

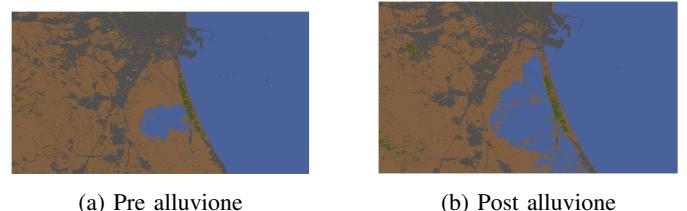


Fig. 11: Comparazione dell'area metropolitana di Valencia

Le predizioni ottenute con il modello da noi allenato sono state successivamente confrontate con i raster NDWI, scaricati sempre dal sito di Copernicus. L'NDWI è un indice utile per individuare e monitorare la presenza di acqua sulla superficie terrestre, anche all'interno di zone con presenza di vegetazione e/o colture. Nelle predizioni del nostro modello, così come nei raster NDWI, è evidente l'aumento di acqua sulla superficie della provincia di Valencia.

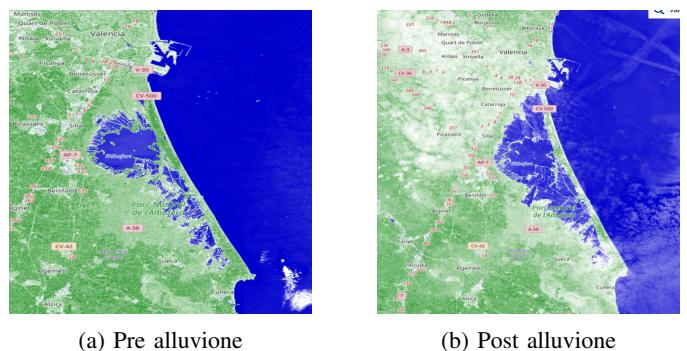


Fig. 12: Raster NDWI zona metropolitana di Valencia

III. GEOMATICA 3D

A. Descrizione del task

Una nuvola di punti è un insieme di punti rappresentati in uno spazio tridimensionale. I dati per la generazione della nuvola vengono catturati da strumentazioni di tipo LiDAR oppure da normali fotocamere facendo dei video, che con appositi algoritmi vengono convertiti in nuvole di punti. I dati delle nuvole di punti possono essere salvati in diversi formati, che in ogni caso contengono le coordinate del punto nello spazio riferite ad un sistema di riferimento, i valori rgb del punto e le normali. Questo task richiede l'effettuazione di una segmentazione semantica di una nuvola di punti tramite il software Cloud Compare. L'uso di questo software, unito all'uso di uno script su Google colab, permette di allenare un modello di rete neurale che sia in grado di fare la predizione della classe di ogni punto costitutente la nuvola. Questo tipo di tecnologia può essere applicato in vari campi, dei quali un esempio è fornito nell'approfondimento 3d che si può trovare nella sezione successiva.

B. Descrizione del dataset

In classe abbiamo avuto modo di imparare ad utilizzare Cloud Compare con un'esercitazione, basata su una nuvola di punti catturata tramite drone del comune di Mappano fornita dalla docente. La nuvola di punti conteneva già alcune features e il risultato della classificazione effettuato in precedenza su Metashape; per poter fare l'esercitazione è stato necessario eliminare alcune di queste informazioni. In particolare, si è scelto di tenere la intensity e le normali.



Fig. 13: Nuvola da classificare

Prima di effettuare l'allenamento della rete abbiamo scelto di calcolare delle features per facilitare classificazione dei vari punti da parte del nostro modello. Innanzitutto, abbiamo calcolato la **Verticality**, che si riferisce alla distribuzione dei punti lungo la direzione verticale; abbiamo utilizzato un raggio della sfera di 10 cm.

Successivamente abbiamo calcolato l'**Omnivariance** con un raggio di 0.2; questa feature ci identifica quanto è uniforme la nuvola di punti; ha infatti un valore più alto negli alberi e nelle finestre(dove i punti sono distribuiti in più direzioni), e

un valore basso sui tetti o sulle strade.

Come ultima feature abbiamo calcolato la **planarity** con un raggio di 0.6; questa feature si riferisce alla distribuzione dei punti lungo la direzione orizzontale.

I punti del dataset possono appartenere ad una delle seguenti classi:

- 0) Building - Blu
- 1) Vegetation - Azzurro
- 2) Car - Verde
- 3) Street - Verde
- 4) Ground - Giallo
- 5) Dividing element - Arancione
- 6) Other - Rosso



(a) Nuvola di Training

(b) Nuvola di Validation

C. Modello utilizzato

Per la classificazione della nuvola di punti è stato utilizzato un algoritmo RandomForestClassifier. Per ogni tentativo la rete è stata allenata con il Random Forest eseguito con il criterio di splitting gini, profondità 50 e tre diversi valori di estimatori (100, 150 e 200). Per quanto riguarda il Gini index, esso fa riferimento ad una misura di impurità del nodo usata per determinare qual è il miglior modo di dividere i dati in un albero decisionale. Per profondità si intende quella raggiungibile da ogni singolo albero decisionale elaborato dall'algoritmo di random forest durante la fase di allenamento. Nella sezione successiva, contenente l'analisi dei risultati ottenuti abbiamo riportato la configurazione di parametri con i quali abbiamo avuto i risultati migliori.

D. Analisi delle performance del modello

La nuvola di punti è stata divisa in 3 dataset: uno di training, uno di validazione e un terzo per fare il test del modello. Per arrivare a un risultato soddisfacente sono stati necessari tre tentativi. Il primo tentativo è stato eseguito per tutte le features del dataset ricevuto ed ha avuto un'accuracy pari a zero per la classe other perché gli elementi di questa classe sono stati erroneamente annotati come "ground" nel dataset di training fornito all'algoritmo. In seguito ad un'analisi del nostro lavoro abbiamo capito che erano stati commessi due ulteriori errori. In primis, abbiamo utilizzato un campione troppo piccolo per il training e il validation della classe vegetation, utilizzando soltanto alberi verdi.

Il secondo errore è stato quello di testare il nostro modello sulla nuvola originale (comprendente anche dei punti utilizzati per il training e validation). Agendo in questa maniera, i

risultati vengono falsati in quanto il modello conosce già una porzione della nuvola di punti sulla quale viene testato.

Class	Precision	Recall	F1-Score	Support
0	0.83	0.87	0.85	10,996
1	0.99	0.90	0.95	24,191
2	0.76	0.55	0.64	1,497
3	0.93	0.89	0.91	14,570
4	0.72	1.00	0.84	9,719
5	0.87	0.59	0.70	1,744
6	0.00	0.00	0.00	435
Accuracy	0.89			
Time needed	6.25 minuti			

TABLE II: Risultati del primo tentativo, iperparametri Gini 150 50

Il secondo tentativo è stato fatto considerando tutte le features calcolate, arrivando ad un'accuratezza di 0.84. Abbiamo però commesso l'errore di non considerare i bidoni della raccolta differenziata gialli nella classe other. Infatti la precision è solamente 0.43.

Class	Precision	Recall	F1-Score	Support
0	0.87	0.98	0.92	12,992
1	0.97	0.84	0.90	42,636
2	0.22	0.24	0.23	1,854
3	0.96	0.80	0.87	14,999
4	0.58	1.00	0.73	12,068
5	0.84	0.40	0.55	3,987
6	0.43	0.40	0.41	228
Accuracy	0.84			
Time needed	7.48 min			

TABLE III: Secondo tentativo effettuato con iperparametri gini 150 50, considerando tutte le features

Il terzo tentativo è stato eseguito comprendendo i bidoni nel dataset di training, infatti la precision di other migliora aumentando a 0.60. Inoltre abbiamo svolto i test sia considerando solamente i colori dei punti e relative normali sia considerando tutte le features. Abbiamo ottenuto un risultato di accuracy di 0.84 per tutte le features e 0.57 per il training solo RGB e normali. Per quanto riguarda i tempi di allenamento della rete, come ci aspettavamo il training considerando tutte le features richiede più tempo (poco più del doppio).

Class	Precision	Recall	F1-Score	Support
0	0.87	0.98	0.92	12,992
1	0.97	0.84	0.90	42,636
2	0.23	0.25	0.24	1,854
3	0.96	0.77	0.85	14,999
4	0.57	1.00	0.73	12,068
5	0.84	0.41	0.55	3,987
6	0.60	0.90	0.72	288
Accuracy	0.84			
Time needed	7.52 minuti			

TABLE IV: Terzo tentativo effettuato con iperparametri gini 150 50, considerando tutte le features

Class	Precision	Recall	F1-Score	Support
0	0.62	0.98	0.76	12,992
1	0.88	0.38	0.53	42,636
2	0.30	0.20	0.24	1,854
3	0.86	0.62	0.72	14,999
4	0.33	0.84	0.48	12,068
5	0.23	0.28	0.25	3,987
6	0.10	0.95	0.18	288
Accuracy	0.57			
Time needed	3.08 minuti			

TABLE V: Risultati terzo tentativo effettuato con iperparametri gini 200 50, considerando solo rgb e normali

Risulta evidente come, considerando tutte le features, il modello sia in grado di sviluppare una buona capacità di predizione. La performance di accuracy ottenuta considerando rgb (57%), dimostra come le altre features (omnivariance, verticality, planarity e normali) siano fondamentali per predire l'appartenenza ad una classe. Questo si nota particolarmente per le classi Building, Dividing element e Other, dove la precision cala drasticamente. Per quanto riguarda le altre classi, invece, le features rgb aiutano in maniera maggiore la rete neurale a sviluppare predizioni.

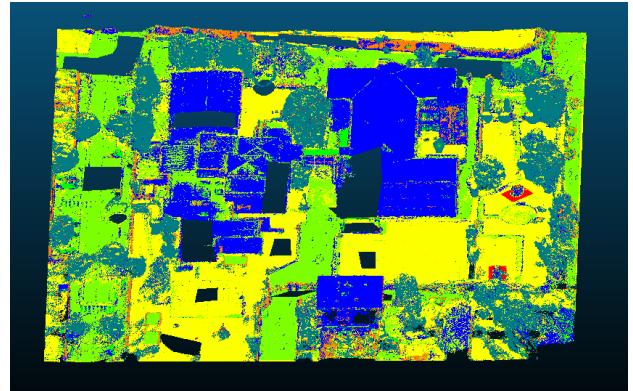


Fig. 15: Predizione finale del modello migliore

Il modello ha commesso qualche errore di predizione, in particolare sulle macchine, sugli edifici e sugli elementi "Other". Questo potrebbe essere causato da una mancanza di punti in certe classi nel dataset di training. Ci riteniamo comunque soddisfatti della predizione ottenuta.

IV. APPROFONDIMENTO 3D

Questo task consente di approfondire ulteriormente la segmentazione 3d grazie all'uso di Cloud Compare. In questo caso, ci siamo focalizzati sulla segmentazione semantica di nuvole di punti riguardanti elementi del patrimonio culturale e architettonico. La classificazione semantica dei dati 3D del patrimonio culturale può aiutare la comunità a comprendere e analizzare meglio i gemelli digitali delle infrastrutture e facilitare i lavori di restauro e conservazione[1]. È stato utilizzato ArCH dataset, composto da diverse nuvole di punti; nel nostro caso è stata utilizzata la nuvola presente nel file denominato

“Training/1_TR_cloister.txt”. Durante l’importazione della nuvola è stato necessario fare attenzione al formato con cui è stata salvata, il quale non corrisponde allo standard di Cloud Compare; è stato necessario perciò adattare le normali ed il valore scalare che indica la classe a cui appartiene il punto.

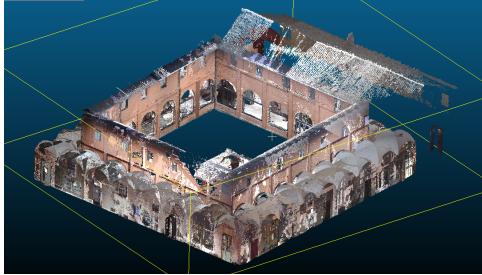


Fig. 16: Nuvola di punti

La nuvola di punti è stata suddivisa in 3 set: un gruppo di punti per il training dei dati, uno per la validation ed infine i dati restanti per effettuare il testing.

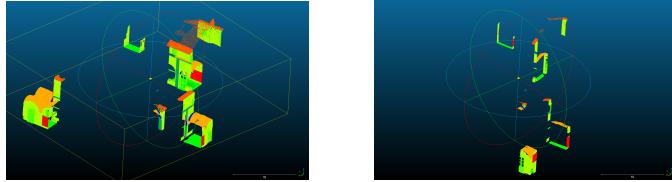


Fig. 17: Elementi utilizzati per l’allenamento del modello

La prima suddivisione in 3 set che è stata fatta non aveva un numero sufficiente di dati per poter raggiungere dei buoni livelli di performance. Oltre ad aumentare la quantità di dati inseriti nel set di training, sono state aggiunte altre features, quali omnivariance, planarità e verticalità. Le features aggiunte sono state calcolate con raggio rispettivo di 0.2, 0.4 e 0.2.

I dati sono annotati nelle seguenti classi:

- 0) Archi
- 1) Colonne
- 2) Modanature
- 3) Pavimenti
- 4) Porte e Finestre
- 5) Muri
- 6) Scale
- 7) Volte
- 8) Tetti
- 9) Altro – Elementi non classificabili nelle classi precedenti

Per la classificazione della nuvola di punti è stato utilizzato un algoritmo RandomForestClassifier, come usato anche nell’esercitazione in classe. L’algoritmo è stato eseguito con il criterio gini, 100 estimatori e profondità massima 25. Sul motore Google Colab con più di 100 estimatori o una profondità maggiore, il processo veniva interrotto in quanto occupava troppa RAM sul sistema.

Dopo circa 90 minuti di training del modello abbiamo ottenuto i risultati elencati in questa tabella, In particolare: accuracy = 0.59.

	precision	recall	f1-score	support
0	0.38	0.12	0.18	19439
2	0.17	0.05	0.08	23446
3	0.81	0.85	0.83	13774
4	0.39	0.35	0.37	23590
5	0.68	0.80	0.74	147984
6	0.00	0.00	0.00	0
7	0.71	0.52	0.60	46331
8	0.49	0.72	0.58	13990
9	0.05	0.12	0.07	10575
accuracy			0.59	299129
macro avg	0.41	0.39	0.38	299129
weighted avg	0.58	0.59	0.57	299129

Fig. 18: Risultati primo tentativo

Con il dataset di training più grande rispetto al primo tentativo, e le features aggiunte come sopra indicato, abbiamo un’accuracy di 0.64 (leggermente migliore del tentativo precedente).

	precision	recall	f1-score	support
0	0.57	0.18	0.27	19439
2	0.39	0.29	0.33	23446
3	0.81	0.87	0.84	13774
4	0.29	0.41	0.34	23590
5	0.74	0.78	0.76	147984
7	0.66	0.75	0.70	46331
8	0.71	0.63	0.67	13990
9	0.06	0.03	0.04	10575
accuracy			0.64	299129
macro avg	0.53	0.49	0.49	299129
weighted avg	0.63	0.64	0.63	299129

Fig. 19: Risultati secondo tentativo

Al riguardo dell’assenza dei risultati riguardanti le classi 1 e 6, questo si è verificato in quanto queste classi non sono presenti in questa sequenza che abbiamo analizzato. Si allega qua sotto la predizione finale.

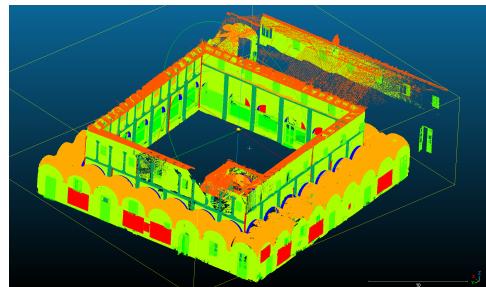


Fig. 20: Nuvola di punti

V. INFORMATICA 2D

A. Descrizione e Motivazioni del Task

Nella classificazione semplice, si cerca di classificare un oggetto in primo piano contenuto nell’immagine, tuttavia questo approccio è poco funzionale nelle applicazioni reali. È infatti improbabile, eccetto per task specifici, dover analizzare delle immagini con oggetti in primo piano che rappresentano l’intera immagine.

La segmentazione semantica estende il task di classificazione semplice ad immagini più complesse contenenti più oggetti e categorie all'interno della stessa scena. Questo è possibile andando a classificare ogni singolo pixel dell'immagine. Grazie a questo approccio viene fornita una comprensione visiva molto più dettagliata della scena rispetto alla classificazione semplice. Consente, infatti, di conoscere la forma e la posizione esatta di ogni classe.

Le applicazioni sono molteplici e importanti per diversi settori tra cui la guida autonoma, dove l'identificazione in tempo reale di oggetti e persone migliora la sicurezza e l'affidabilità dei sistemi di navigazione autonoma.

Inoltre, un'applicazione degna di nota la si trova nel campo geospatiale, dove la classificazione semantica permette una semplificazione dell'analisi delle immagini satellitari, permettendo di effettuare stime, previsioni e mappature. Per esempio, facilita l'individuazione e la previsione di frane, incendi ed altre catastrofi naturali, la stima di bacini idrici, foreste e terreni agricoli, supportando la pianificazione urbana e la gestione delle risorse naturali.

B. Descrizione del Dataset

Per questo task utilizzeremo il Dataset LoveDA[2], che include 5.987 immagini ad alta risoluzione (HSR) e 166.768 oggetti annotati, raccolti da tre città diverse. LoveDA presenta due domini distinti, urbano e rurale. Ciò introduce una difficoltà a causa della presenza di oggetti di dimensioni variabili, sfondo complesso e distribuzione delle classi non uniforme. Il Dataset è composto dalle seguenti classi: **Background**, **Building**, **Road**, **Water**, **Barren**, **Forest**, **Agriculture**.

Per l'addestramento della rete neurale utilizzeremo il dominio urbano, composto rispettivamente da 1156 immagini per il training e 677 per il validation.

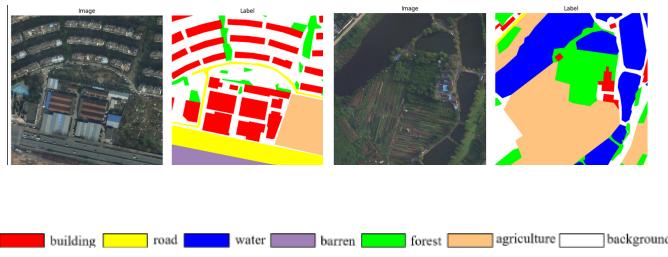
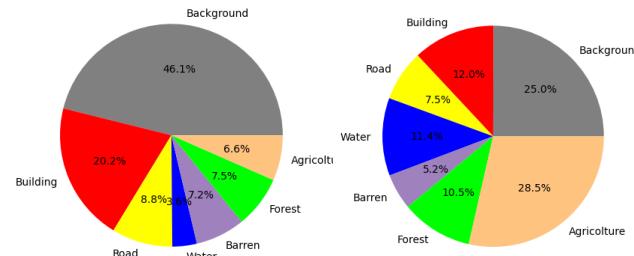


Fig. 22: Esempio di immagini e legenda delle classi

La data imbalance è lo squilibrio delle varie classi all'interno del dataset. Nel nostro caso, utilizzando solamente il dominio urbano, è presente in maggioranza la classe Background mentre la classe Agriculture è la meno presente.

Com'è possibile notare dai grafici (Figura 23), la classe Background è predominante, mentre la classe Agriculture è quasi inesistente all'interno del Training Dataset, ma molto presente nel Validation. Questo crea delle problematiche nel training della rete, tra le quali una tendenza a favorire le classi più rappresentate, riducendo la capacità di predire correttamente quelle meno presenti. Per risolvere il problema si possono applicare tecniche di data augmentation, ad esempio



(a) Distribuzione classi nel dataset di training
(b) Distribuzione classi nel dataset di validation

Fig. 23: Data imbalance

aggiungendo al dataset immagini contenenti in misura preponderante le classi meno rappresentate. Le immagini aggiunte si possono ottenere mediante trasformazioni applicate alle immagini di partenza.

C. Modello Utilizzato

Il modello che abbiamo utilizzato si chiama BiseNet (Bilateral Segmentation Network)[3].

A differenza di reti di classificazione semplice, nel nostro caso è importante preservare le informazioni a livello di singolo pixel, mantenendo così la risoluzione spaziale dell'immagine originale. La BiseNet implementa una struttura a due percorsi in modo da mantenere la massima risoluzione spaziale senza perdere informazioni dal contesto. Il primo path, chiamato **Spatial Branch** è composto da soli tre layer convoluzionali, i quali riducono la risoluzione fino a un ottavo. L'altro path, chiamato **Contextual Branch**, contiene più layer convoluzionali arrivando a ridurre la risoluzione fino a un trentaduesimo, in modo da diminuire di molto la risoluzione spaziale, ma fornendo molto contesto. Alla fine del Context Path viene fatto un Global Average Pooling in modo da raccogliere le informazioni globali.

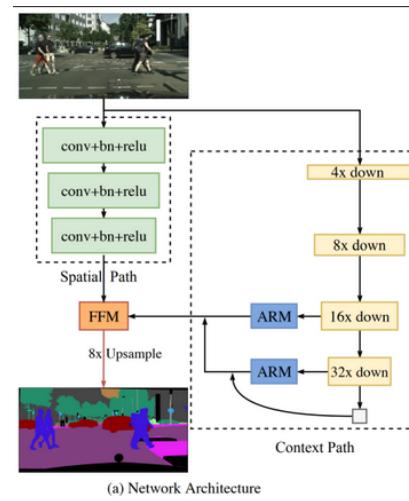


Fig. 24: I due path della BiseNet

L'output finale della rete è un'immagine della stessa dimensione dell'input, ma con un numero di canali pari al numero di classi. Ogni canale rappresenta la probabilità che un dato pixel appartenga a una specifica classe. Per ogni pixel viene dunque prodotto un vettore di probabilità, la più alta delle quali corrisponde alla classificazione del pixel stesso.

D. Analisi delle Performance del Modello

Per trovare gli iperparametri ideali per il training della nostra rete abbiamo fatto vari tentativi.

Abbiamo inizialmente commesso l'errore di utilizzare un learning rate troppo alto (10^{-3}) per il batch size utilizzato (16), durante una fase di addestramento del modello per 10 epoch (Fig. 25). L'errore commesso non permette alla rete di raggiungere un minimo locale, facendo incrementare la Loss dopo il raggiungimento di un plateau; con questa configurazione abbiamo raggiunto un valore minimo della Loss di 3.48 e una mIoU massima di 0.22. La mIoU (Mean Intersection over Union) è il rapporto tra l'area sovrapposta e l'area dell'unione (considerando rispettivamente la predizione del nostro modello e la ground truth. Questo dato è importante perché fornisce una metrifica di accuratezza della predizione.

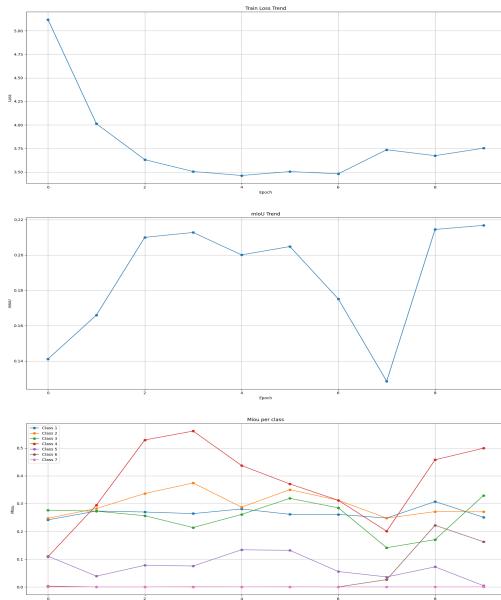


Fig. 25: Trend di Loss e mIoU tentativo 1

Successivamente abbiamo quindi provato ad allenare la rete per più epoch (20) utilizzando uno Step Size di 15 con parametro GAMMA di 0.1. Lo Step Size è un parametro dello Scheduler StepLR che diminuisce il Learning Rate moltiplicandolo per il parametro GAMMA ogni numero di epoch specificato. Abbiamo però notato che la Loss, dopo una minima discesa iniziale, ha raggiunto subito un plateau a 3.0,

per poi iniziare a crescere. La mIoU, dopo un iniziale aumento, ha iniziato a decrescere (Fig. 26). Il valore massimo raggiunto è stato di 0.24 nella quindicesima epoch di addestramento.

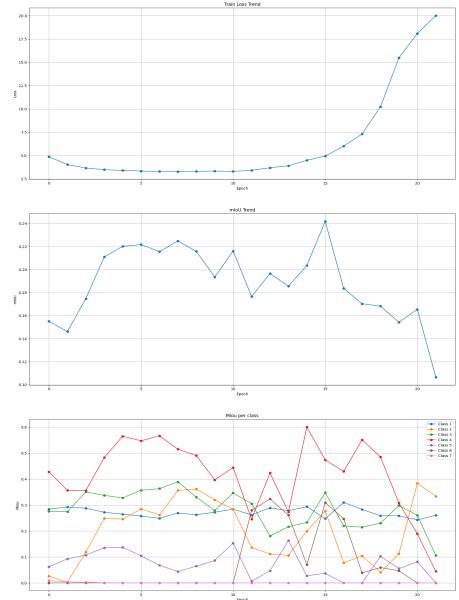


Fig. 26: Trend di Loss e mIoU tentativo 2

Abbiamo infine provato a rimuovere lo Step Size e diminuire il Learning Rate a 10^{-4} ; con questi parametri siamo riusciti ad avere un trend sempre decrescente della Loss Function (nessun plateau), fino a raggiungere il valore di 1.3 e un buon valore della mIoU (0.41).

Dopo aver trovato dei parametri che ci restituivano dei valori soddisfacenti, abbiamo infine provato a cambiare anche l'optimizer da SGD ad Adam. L'optimizer è un elemento fondamentale della rete, in quanto è la funzione responsabile del cambiamento e dell'ottimizzazione degli iperparametri durante la fase di training. Fornendo i parametri della rete e il Learning Rate all'optimizer, esso aggiorna i parametri del modello ad ogni epoch, in base al learning rate e al gradiente calcolato durante la backpropagation. Con il cambiamento dell'optimizer, abbiamo notato che la Loss function diminuiva molto più velocemente, e la mIoU arrivava a valori più alti. Abbiamo dunque concluso che l'optimizer Adam è più veloce nel raggiungimento di un minimo della Loss Function; lo consideriamo quindi più adatto al nostro task considerate le limitate risorse di computazione. Così facendo abbiamo ottenuto un buon trend decrescente della Loss (Fig. 27), arrivando ad una mIoU di 0.42 (decisamente maggiore rispetto ai test precedenti).



Fig. 27: Trend di Loss e mIoU finali

Come ultimo tentativo abbiamo implementato delle tecniche di Data Augmentation sul dataset di training. In particolare abbiamo provato ad applicare delle trasformazioni quali Horizontal e Vertical flip, su dei patches casuali di 512x512 pixel all'interno delle immagini, con probabilità del 20% e del 50% (Fig. 28). Applicando tecniche di Data Augmentation simili si evita che il modello impari a memoria le features delle immagini di input, aumentando la generalizzazione delle conoscenze acquisite dalla rete e dunque riducendo la probabilità di overfitting.

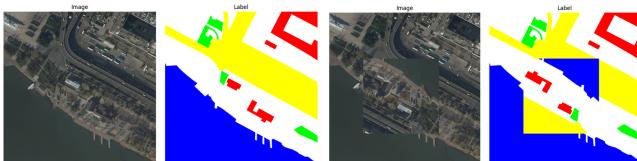


Fig. 28: Esempio di Data Augmentation

Tuttavia, non ci sono stati miglioramenti evidenti del modello. Infatti la mIoU ottenuta è stata di 0.35 per l'augmentation con la probabilità del 20% e 0.275 con la probabilità del 50%.

E. Risultati Qualitativi

Come risultato finale abbiamo ottenuto una mIoU di 0.42. La performance del modello è certamente influenzata negativamente dal forte sbilanciamento del Dataset, sfavorendo la predizione di alcune classi. Sicuramente la Mean Intersection over Union potrebbe crescere ancora continuando il training con altre iterazioni, dato che la Loss function non aveva ancora raggiunto un minimo. Ci riteniamo soddisfatti del risultato finale; sono riportati due esempi di predizione del nostro modello.

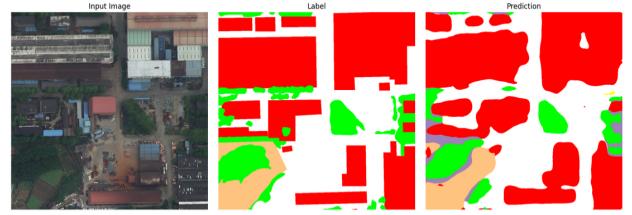


Fig. 29: Prima predizione

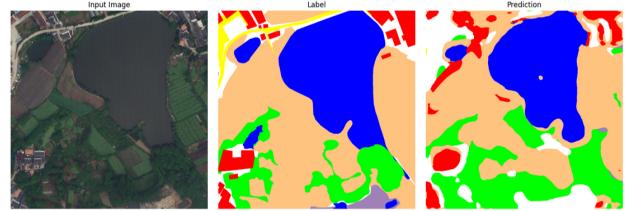


Fig. 30: Seconda predizione

VI. INFORMATICA 3D

A. Descrizione del task

In questo caso il task richiede di classificare elementi tridimensionali. A differenza dei dati 2d, costituiti da immagini (e dunque da singoli pixel), la realtà viene rappresentata in tre dimensioni tramite un nuovo tipo di dato: la point cloud. Lo spazio viene modellizzato tramite un set non ordinato di punti, ognuno dei quali è associato alla propria posizione nello spazio (coordinate x,y e z). A seconda di dove si trovino e/o di quale oggetto reale rappresentino, questi punti possono essere classificati seguendo una lista di categorie (dette classi). Questo ci permette di avere una piena comprensione della realtà osservata, definendo le forme che la compongono. La segmentazione di dati tridimensionali e la creazione di mappe di segmentazione svolgono un ruolo importante nell'addestrare i computer a riconoscere un contesto significativo in ambienti reali come paesaggi naturali, ambienti urbani e interni di edifici. Le possibili applicazioni industriali di questa tecnica spaziano in un ampio spettro di attività che variano dalla guida autonoma all'ambito medico nella ricerca di tumori cerebrali[4].

B. Descrizione del dataset

Per effettuare la segmentazione semantica si è scelto di usare il dataset SemanticKITTI[5]. È basato sul dataset odometry di KITTI, che raccoglie dati provenienti ambienti urbani attinenti alla città di Kalsruhe, in Germania. SemanticKITTI include oltre 43.000 scansioni LiDAR annotate con 28 classi. Le classi sono state scelte per coprire un'ampia varietà di oggetti, incluse categorie generiche come "other-structure" o "other-vehicle", pensate per casi ambigui. Una classe outlier è inclusa per gestire errori di misurazione dovuti a riflessi o limitazioni del sensore. Questo dataset contiene dati grezzi e annotati su 22 sequenze, con circa 23.000 scansioni per l'addestramento e 20.000 per la fase di test.

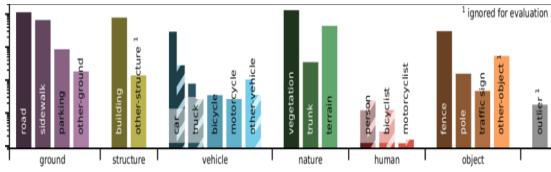


Fig. 31: Classes of the dataset

Seguendo il paper, abbiamo scelto di trainare di una rete di 3D Semantic Segmentation su 26 classi. Nello specifico, le classi other-ground e other-structure sono state mappate ad outlier. Inoltre, per limitare l'uso di risorse GPU, non utilizzeremo l'intero dataset, ma un sample di alcune sequenze significative per il training (sequenze 0, 2 e 3) e per la validation del modello (sequenza 1). Un parametro importante per le sequenze contenute nel dataset è la scelta del numero di punti contenuti in ogni cloud. Riportiamo qua sotto un esempio riguardante il cambiamento di aspetto di una nuvola al variare del numero di punti. In aggiunta a ciò, bisogna tenere in considerazione che le performance della rete che andremo ad addestrare potrebbero variare leggermente al cambiamento del numero di punti per singola cloud (vedasi sezione 'Analisi delle performance del modello').

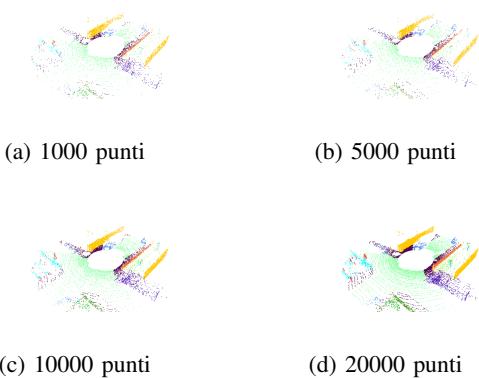


Fig. 32: Variazione di aspetto delle point cloud al variare del numero di punti

Come è evidente da questi esempi, l'aumento del numero di punti consente una migliore comprensibilità della scena rappresentata.

C. Modello utilizzato

Il modello che abbiamo utilizzato per la segmentazione semantica delle point cloud 3d è chiamato PointNet[6]. Si tratta di una rete neurale progettata per elaborare direttamente point cloud 3D non strutturate, senza la necessità di convertirle in voxel o mesh.

La rete di classificazione prende in input n punti, applica trasformazioni sull'input e sulle feature di questi punti (tramite l'uso di multi layer perceptron). In seguito, aggrega le caratteristiche dei punti mediante max pooling. L'output è costituito

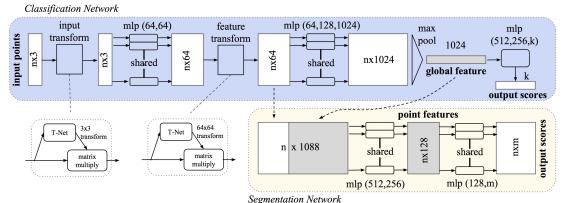


Fig. 33: PointNet

dai punteggi di classificazione per k classi (nel nostro caso k=26). La rete di segmentazione è una successiva estensione della rete di classificazione. Nel concreto, essa concatena le caratteristiche globali e locali e produce un punteggio per ogni punto (anche in questo caso grazie all'uso di percetroni multistrato).

Questa rete consente un approccio innovativo per vari task di 3d recognition, tra i quali classificazione di oggetti, segmentazione di parti e segmentazione semantica.

D. Analisi delle performance del modello

Per addestrare il modello in maniera efficace abbiamo effettuato vari tentativi per stabilire il set migliore di iperparametri. Il primo su cui ci siamo concentrati, anche come conseguenza dei risultati ottenuti nella segmentazione 2d, è stata la scelta dell'optimizer. In particolare, considerate le ancor più limitate capacità di computazione per la fase di training e validation (in questo caso un'epoca dura circa 18-20 minuti) abbiamo scelto di utilizzare Adam (Adaptive Moment Estimation) perché, come abbiamo scoperto durante il task di segmentazione semantica 2d, si muove velocemente verso il minimo della loss function.

Un altro iperparametro importante è il learning rate, per il quale abbiamo scelto di usare il valore di (10^{-3}) . Per quanto riguarda il batch size, invece, siamo stati costretti a usare 8, in quanto fissando questo iperparametro a 16 la GPU fornita da Google Colab esauriva la memoria e non ci permetteva dunque di effettuare il training e la validation del modello.

Un altro parametro su cui ci siamo focalizzati inizialmente è il numero di punti per nuvola. La nostra aspettativa era quella di una maggiore capacità del modello di acquisire conoscenza all'aumentare del numero di punti per nuvola (e di conseguenza un aumento della grandezza e della variabilità del dataset usato per il training). Per fare un veloce confronto abbiamo fatto dei rapidi test di 3 epochi del modello con 2000 punti per nuvola e 5000 punti per nuvola (Fig. 34).

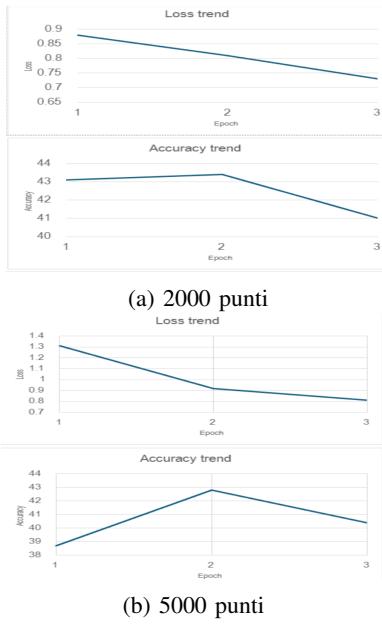


Fig. 34: Comparazione 2000 e 5000 punti

A seguito di questi risultati, abbiamo optato per una configurazione con 5000 punti per nuvola in quanto, pur ottenendo una percentuale di accuracy massima simile (43.4 con 2000 punti e 42.8 con 5000 punti), il modello si muoveva molto più velocemente verso il minimo del gradiente della loss function (è scesa in sole tre epocha da 1.32 a 0.81).

Terminata questa fase di analisi e di scelta dei valori da assegnare agli iperparametri abbiamo fatto un test di 7 epocha per vedere quali percentuali di accuracy avremmo ottenuto.

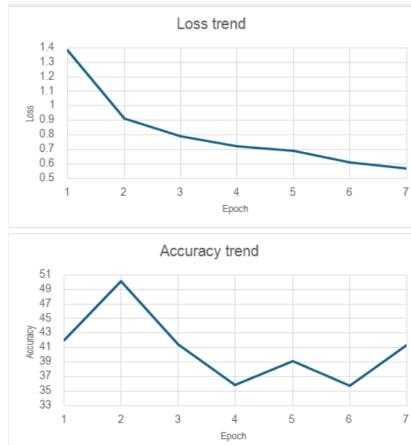


Fig. 35: 5000 punti, 7 epocha

Il risultato di questo test (Fig. 35) è stato decisamente lontano da quel che ci aspettavamo. Il valore di accuracy di poco superiore al 50%, unito al fatto che sia stato ottenuto durantela seconda epoca di allenamento, ci ha portato a dover fare alcune riflessioni. In primis, è interessante notare come, di fronte a una loss sempre decrescente, la accuracy ha un andamento

decisamente discontinuo. La nostra ipotesi per spiegare questo fenomeno si fonda su due fattori principalmente:

- Possibile data imbalance tra le sequenze usate per il training e le sequenze usate per il validation
- Overfitting dei dati di training

Introdurre tecniche di data augmentation potrebbe essere utile per ridurre l'overfitting e potenzialmente mitigare la data imbalance.

E. Data augmentation

Abbiamo scelto di testare due tecniche di data augmentation:

- Point Jitter
- Point dropout

Il **point jitter** consiste nell'applicare un rumore gaussiano $N(0,\sigma)$ alle coordinate (x,y,z) dei punti nella point cloud. Il rumore viene applicato solo ad alcuni batch in maniera randomica (con una probabilità del 20% per ogni batch). Per questi tentativi abbiamo scelto la configurazione del dataset con 5000 punti per nuvola e abbiamo testato due valori di varianza del rumore gaussiano per un totale di 5 epocha di training per ciascun test:

- $\sigma = 0.01$
- $\sigma = 0.05$

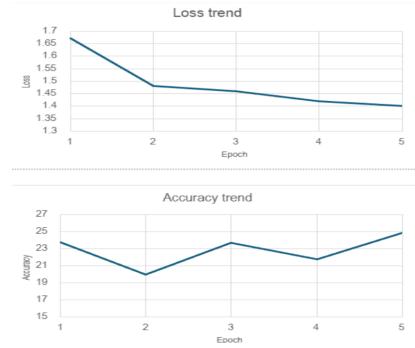


Fig. 36: Jitter con rumore gaussiano $N(0,0.01)$

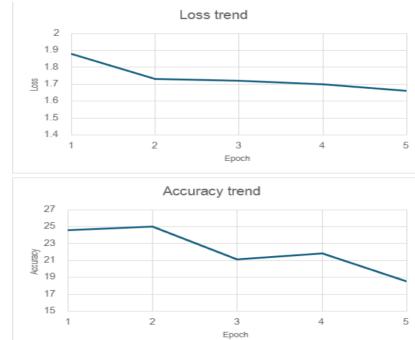


Fig. 37: Jitter con rumore gaussiano $N(0,0.05)$

Entrambi i test ci hanno fornito risultati decisamente peggiori rispetto ai test effettuati in precedenza senza l'uso di tecniche di data augmentation. Un tratto comune è sicuramente

la lieve discesa della loss a fronte di un'accuracy molto bassa (circa il 20% in meno rispetto ai test iniziali). Una prima ipotesi è che la percentuale di batch che subiscono l'aggiunta del rumore gaussiano scelta è errata, tuttavia non è sufficiente a spiegare un'abbassamento della accuracy così netto. Molto probabilmente, la tecnica di jitter è stata applicata in maniera imprecisa ed è per questo motivo che i risultati non sono in linea con gli effetti teorici di una tecnica di data augmentation.

Il **point dropout** consiste nell'applicare una maschera a una percentuale casuale di punti (detta *dropout percentage*) nella point cloud, sostituendo le loro coordinate con (0,0,0). A differenza della tecnica precedentemente approfondita, il dropout si applica all'intero dataset prima che esso inizi ogni epoca di training. Anche in questo caso la configurazione scelta per il dataset è quella con 5000 punti per point cloud e un numero di epoche di training pari a 5. Per testare l'efficacia di questa tecnica abbiamo scelto due valori di *Dropout percentage*:

- 10%
- 30%

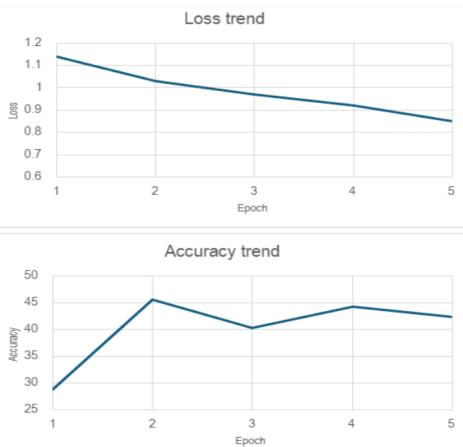


Fig. 38: Dropout percentage = 10%

Risulta molto evidente come le performance ottenute con una percentuale di dropout del 10% sono decisamente migliori per quanto riguarda il valore della loss. Essa ha un ottimo trend e giunge fino a un valore di poco inferiore a 0.9 (ben 0.2 più basso rispetto al test con dropout percentage = 30%). Se invece ci concentriamo sulla accuracy, entrambi i test hanno raggiunto un risultato di circa 45 %, sebbene con una dropout percentage di 0.3 il modello abbia impiegato più epocha a raggiungere questa performance. L'ipotesi che riteniamo più probabile per spiegare l'andamento diverso delle performance, anche tenendo conto della conclusione sull'overfitting ottenuta dal training della PointNet senza data augmentation, è che all'aumentare della dropout percentage il modello ci mette più tempo a avvicinarsi a un minimo locale della funzione di loss. Per concludere, entrambe le tecniche non hanno portato a significativi miglioramenti nelle performance del modello rispetto ai casi in cui la data augmentation era assente.

REFERENCES

- [1] Matrone, F., Lingua, A., Pierdicca, R., Malinverni, E. S., Paolanti, M., Grilli, E., Remondino, F., Murtyoso, A., and Landes, T.: A BENCHMARK FOR LARGE-SCALE HERITAGE POINT CLOUD SEMANTIC SEGMENTATION, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLIII-B2-2020, 1419–1426, <https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-1419-2020>, 2020.
- [2] Junjue Wang, Zhuo Zheng, Ailong Ma, Xiaoyan Lu, Yanfei Zhong LoveDA: A Remote Sensing Land-Cover Dataset for Domain Adaptive Semantic Segmentation NeurIPS 2021 Datasets and Benchmarks Track
- [3] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, Nong Sang BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation. ECCV 2018
- [4] Christian Francesco Russo, *Insieme di approcci per la segmentazione 3d del tumore cerebrale*, Università degli Studi di Padova, 2023. Disponibile online: https://thesis.unipd.it/retrieve/f428fdb-53d1-485d-81af-21a35e62a816/Russo_Christian%20Francesco.pdf.
- [5] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, Juergen Gall SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences ICCV2019
- [6] Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation CVPR 2017

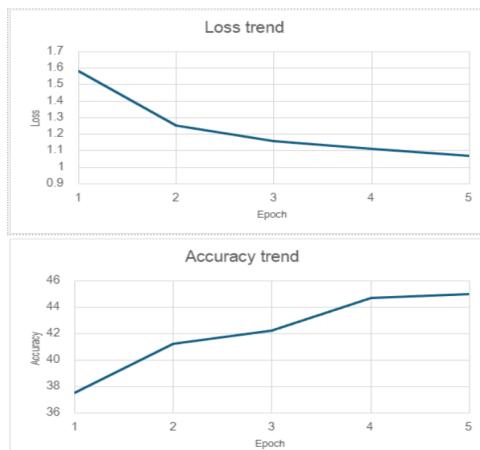


Fig. 39: Dropout percentage = 30%