UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA DELL'INFORMAZIONE

"CLASSIFICAZIONE DI MODELLI 3D CON DEEP LEARNING"

Relatore: Prof. Zanuttigh Pietro

Laureando: Piccoli Marco Annibale

ANNO ACCADEMICO 2021 – 2022

Data di laurea 25/11/2022

Sommario

Abstract

L’obiettivo di questo lavoro è la creazione di un classificatore basato su reti neurali che riesca a distinguere oggetti diversi rappresentati attraverso nuvole di punti. Il modello è basato sull’architettura Res-Net dove il blocco residuo fondamentale è stato preso dalla rete RandLA-net, poiché pensato apposta per l'analisi delle nuvole di punti. I dati, nella forma di liste di coordinate 3D, sono esplorati dal modello con un campo visivo via via più ampio in modo da estrarre proprietà locali e globali dell'oggetto analizzato. Il modello è stato allenato da zero e gli iperparametri sono stati ottimizzati per il task. I risultati della rete sono comparabili con lo stato dell'arte utilizzando il ModelNet10 come test.

Introduzione

Negli ultimi anni l'analisi e l'elaborazione di oggetti e scene tridimensionali sta acquisendo sempre più importanza nel campo della computer vision [1]. Fino a poco tempo fa, l'approccio privilegiato era quello di analizzare dati in due dimensioni provenienti da sensori fotografici (le classiche fotografie), che però contengono solamente delle proiezioni di oggetti tridimensionali, perdendo l'informazione sulla profondità. L'analisi e l'estrapolazione da dati tridimensionali è in grado di conferire ai computer l'abilità di navigare nello spazio attorno a sé e di riconoscere esseri ed oggetti che lo popolano, il tutto grazie a nuove tipologie di sensori in grado di raccogliere dati 3D via via più definiti come i sensori Lidar e le "time of flight depth camera" incluse in alcuni telefoni di ultima generazione. La nuova abbondanza di dati, e di conseguenza il fiorire dei dataset costruiti a partire da questi dati hanno permesso alla comunità scientifica di produrre nuovi modelli di machine learning in grado di processare enormi quantità di dati rapidamente, raggiungendo livelli di accuratezza sempre migliori.

Point Clouds

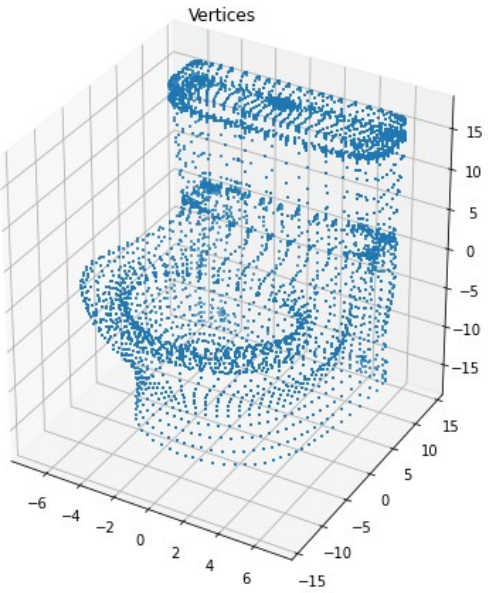

Figura 1: Nuvola di punti estratta dal ModelNet10

I dati analizzati in questo progetto sono detti point cloud, o nuvole di punti. Queste non sono altro che un insieme di punti definiti da tre coordinate ed eventualmente degli ulteriori attributi, che possono essere dati dal sensore che li ha raccolti, ad esempio il colore in formato RGB o possono essere creati artificialmente come ad esempio con il calcolo della curvatura locale della superficie di un oggetto. La natura tridimensionale dei dati e la loro rappresentazione nella memoria dei calcolatori pongono delle difficoltà nella loro analisi che vari modelli tentano di superare con approcci e filosofie diverse. Le principali avversità nel trattare le nuvole di punti sono le seguenti:

- **Irregolarità**: le nuvole di punti sono irregolari, nel senso che i punti presenti in esse non sono campionati a distanze costanti gli uni dagli altri, perciò vi si possono trovare regioni di spazio più dense ed altre meno dense. Questo è dato dalla posizione del sensore che ha raccolto e registrato la nuvola di punti, ma comporta che alcune regioni di spazio hanno una sovrabbondanza di informazione locale, utilissima per la classificazione o la segmentazione, ma altre regioni ne contengono poca, causando ambiguità e portando a molti errori nel momento della classificazione.

Figura 2: Confronto tra zone più e meno dense di una nuvola di punti

- **Dati non strutturati**: le nuvole di punti non hanno una struttura regolare che le contiene. Ogni punto è rilevato indipendentemente dagli altri, ed i punti a lui vicini non si trovano a delle distanze fisse. Tutto questo al contrario di altri formati di dato solitamente analizzati come le fotografie dove i singoli dati vivono in una griglia 2D regolare facile da attraversare per estrapolare proprietà locali da usare nei vari compiti come la classificazione e la segmentazione.

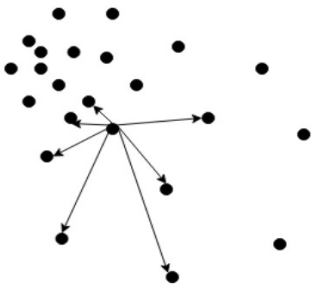

Figura 3: Confronto tra le diverse distanze rispetto ad un pivot point in un cluster di punti

- **Mancanza di ordinamento**: Una nuvola di punti, come definita in precedenza, è solamente un insieme di punti nello spazio, ognuno dei quali con degli eventuali attributi. Le nuvole vengono rappresentate nella memoria dei calcolatori come delle liste o tabelle dove ogni riga rappresenta un singolo punto. Questa rappresentazione introduce delle difficoltà, in quanto l'ordine in cui questi punti è memorizzato non ha importanza, di conseguenza il modello che li analizzerà non dovrà tenere conto della posizione del singolo punto nella tabella, ovvero dovrà essere invariante alla permutazione delle righe nei dati di input.

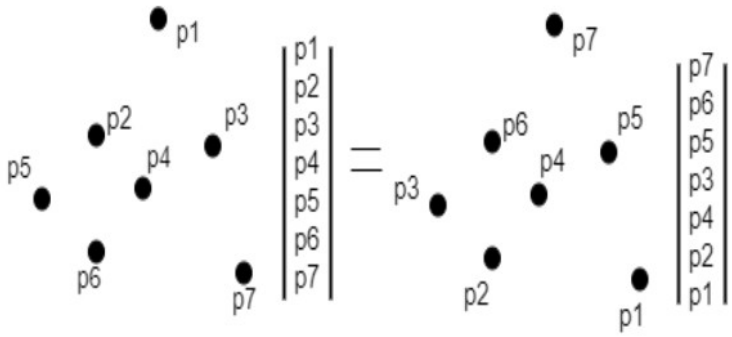

Figura 4: Invarianza alla permutazione delle righe nelle nuvole di punti

Metodi di analisi

Il successo che hanno visto i modelli di deep learning in questi ultimi anni è principalmente basato sull'invenzione del Layer Convoluzionale. Questa tipologia di layer permette alle reti di individuare e determinare dei filtri con cui estrarre delle proprietà dai dati analizzati. L'insieme di queste proprietà individuate e l'importanza della loro presenza o meno vengono poi utilizzati dagli ultimi layer della rete per poter classificare oggetti o scene. Per ovviare alla non strutturalità delle nuvole di punti, molti metodi creano delle strutture artificialmente per poi processare i dati. Queste strutture sono principalmente di tre tipi: immagini 2D, voxel 3D e grafi.

Metodi basati sulla proiezione 2D

Il vantaggio di trasformare una point cloud in un'immagine 2D è che i metodi per l'analisi di queste ultime sono studiati da molti anni ed hanno ottenuto risultati molto soddisfacenti in tutte le loro applicazioni. La trasformazione si ottiene semplicemente proiettando la nuvola di punti su un piano, ottenendo perciò una vista dell'oggetto o scena rappresentato. Da qui poi si procede con i metodi classici per l'analisi, eventualmente aggiungendo più viste da diverse angolazioni della nuvola di punti da analizzare in parallelo, poi da aggregare nel momento della classificazione o segmentazione finale. Per ottenere performance migliori si aggiungono anche proiezioni della nuvola di punti che riportano la curvatura della superficie dell'oggetto oppure la sua densità [2]. Questi modelli raggiungono accuratezze elevate nelle predizioni grazie ai progressi nel campo dell'analisi delle immagini 2D.

Metodi basati su voxelizzazzione

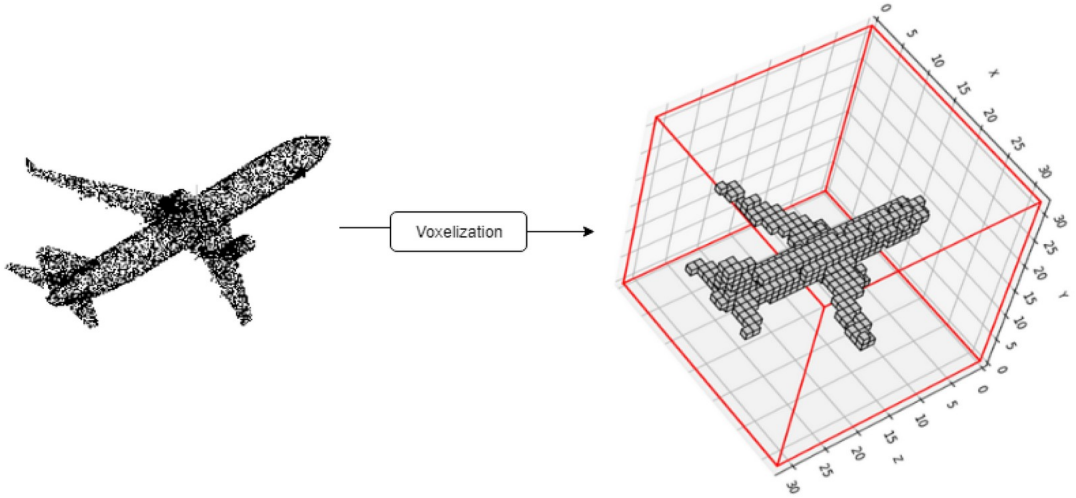

Figura 5: Voxelizzazzione di una nuvola di punti

Un altro metodo molto utilizzato è quello della voxelizzazzione: lo spazio tridimensionale dove vivono le nuvole di punti viene suddiviso in un numero finito di cubi, ognuno dei quali viene marcato come pieno o vuoto se contiene o meno dei punti della nuvola (figura 5). Questo procedimento induce una struttura simile a quella delle immagini, alla quale si possono applicare dei layer di convoluzione 3D per processare l'input. Il problema è che solitamente le nuvole di punti occupano una porzione limitata dello spazio in cui vivono, producendo a sua volta delle voxelizzazioni in cui la maggior parte dei cubi presenti nella suddivisione dello spazio è vuota quindi nulla. Questo comporta che il dato in memoria del calcolatore cresca di dimensionalità in maniera esponenziale pur essendo principalmente composto da cubi vuoti. L'aumento di dimensionalità rallenta l'efficienza del modello che analizza questo tipo di dati e limita fortemente la granularità della discretizzazione dello spazio tridimensionale, ovvero impone un limite inferiore sulla grandezza dei singoli cubi in cui può essere suddiviso lo spazio.

Metodi basati sui grafi

L'ultima trasformazione, quella dei grafi, mira a rappresentare le nuvole di punti come dei grafi in cui ogni punto è un nodo, collegato agli altri punti limitrofi dai lati del grafo. In questi approcci gran parte dell'efficienza dei modelli è data dal metodo scelto per la costruzione del grafo. Per l'analisi vengono successivamente applicate tecniche di convoluzioni applicate ai grafi che mirano ad aggregare le informazioni dai vari nodi e lati per produrre features in grado di guidare la rete nel task finale di classificazione.

Classificazione nel machine learning

L'obiettivo della classificazione nell'ambito del machine learning è di mappare un vettore di input **x** ad una delle K classi discrete C_k con k = 1, ..., K presenti nel dataset di partenza [3]. Nella maggior parte dei casi queste classi sono disgiunte, ovvero ad ogni input corrisponde una ed una sola classe tra le K differenti. Lo spazio in cui vivono i vettori di input è quindi suddiviso in regioni di decisione i cui confini sono chiamati confini di decisione o superfici di confine. Per ottenere una classificazione da un modello di machine learning si possono adottare vari metodi per rappresentare le classi finali con cui discriminare gli oggetti di input. Per un problema di classificazione binaria, quindi con K = 2, è vantaggioso avere come output del modello una variabile t ∈ [0,1] in modo tale che a t = 0 venga assegnata la prima classe ed a t = 1 la seconda. Possiamo interpretare i valori intermedi di t, quelli compresi tra 0 ed 1, come la probabilità di un oggetto di appartenere ad una classe piuttosto che ad un'altra. Per K > 2 è possibile estendere il precedente sistema di codifica delle classi semplicemente prendendo un vettore **t** di K elementi, ognuno dei quali contenuto in [0, 1]. Quindi, nel caso specifico del presente progetto, il vettore target t avrà dimensione K uguale a 10 poiché ci sono 10 classi. Ad esempio, il vettore target per la seconda classe C_2 è: $t = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T$. Di nuovo, è possibile interpretare ogni elemento del vettore t_k come la probabilità che la classe predetta sia la k-esima.

Funzione di Loss

Quando si allena un modello di machine learning è fondamentale definire una funzione di costo o Loss Function, essa infatti rappresenta la distanza tra i valori predetti dal modello e quelli reali. L'obiettivo del training è di minimizzare il risultato della funzione di costo in modo tale da ridurre l'errore della rete e di conseguenza aumentarne l'accuratezza. Per poter utilizzare la rappresentazione delle classi descritta nel paragrafo precedente è necessario che il modello abbia come output il vettore t i cui elementi sono tutti compresi tra 0 ed 1. Per ottenere ciò l'ultimo layer del modello deve essere un Softmax, che restituisce un vettore $\dot{\mathbf{t}}$ di predizioni le cui componenti sono date da

$$\dot{t}(y)_k = \frac{\exp(y_k)}{\sum_{j=1}^{K} \exp(y_j)}$$

con **y** il vettore uscente dal penultimo layer della rete neurale. Il layer softmax ha quindi la funzione di convertire valori appartenenti ai numeri reali in probabilità. Con queste probabilità si misura poi la distanza dal vettore **t** che contiene le probabilità reali descritte nel precedente paragrafo. Questa funzione di loss è chiamata Cross Entropy Loss ed è data dalla seguente formula che è in grado di confrontare le probabilità in uscita dal modello con quelle reali:

$$H(\dot{p}, p) = -\sum_{x \in X} \dot{p}(x) \log(p(x))$$

Discesa del gradiente

Ciò che contraddistingue i modelli di reti neurali e, più in generale, quelli di machine learning, è l'abilità di ricercare dei parametri interni in modo tale da aumentare la precisione e l'accuratezza delle predizioni. Per raggiungere tale obiettivo l'approccio adottato consiste nel trovare il minimo della funzione di costo variando i pesi del modello. Nella sua versione più semplice, la discesa del gradiente consiste nel utilizzare le informazioni derivanti dal gradiente calcolato con la funzione di loss e di aggiornare i pesi della rete attraverso piccoli passi nella direzione negativa dello stesso. In sostanza, ad ogni istante τ ogni matrice di pesi **w** viene aggiornata nel seguente modo

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E\left(w^{(\tau)}\right)$$

dove il parametro η è conosciuto come learning rate ed E(·) la funzione di loss. Dopo ogni aggiornamento dei pesi il gradiente viene valutato nuovamente con i nuovi pesi ed il processo viene ripetuto. Da notare che in questa formulazione basilare occorre valutare la funzione E(·) su tutto il dataset di train, rendendo il procedimento di apprendimento assai lento. Per velocizzarlo si utilizza la discesa stocastica del gradiente, algoritmo ottenuto aggiornando i pesi **w** di volta in volta o in gruppi di numero stabilito chiamati batch, non solo dopo aver visto ogni elemento presente nel dataset di training:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n\left(w^{(\tau)}\right)$$

con $E(w) = \sum_{n=1}^{N} E_n(w)$.

Backpropagation

Per calcolare in maniera efficiente il gradiente di una loss function la tecnica più adottata si basa su uno schema per il passaggio locale delle informazioni, dove le informazioni sul gradiente sono passate in entrambe le direzioni tra il layer che compongono la rete. Questo processo viene denominato backpropagation. Nel primo stadio della backpropagation si calcolano le derivate della funzione di costo rispetto ai singoli pesi $\mathbf{w_i}$ presenti in ogni layer del modello a partire dall'ultimo arrivando al primo grazie alla regola della catena. Nel secondo stadio le derivate vengono utilizzate per calcolare gli aggiornamenti da applicare ai singoli pesi $\mathbf{w_i}$.

ModelNet10 dataset

Il ModelNet10 dataset è un sottoinsieme del ModelNet40 creato dall'Università di Princeton del New Jersey contenente 4899 oggetti appartenenti a 10 diverse categorie [4]. Il dataset è diviso in 3991 oggetti per il training (80%) e 908 oggetti per la fase di test (20%). Tutti gli oggetti provengono da dei modelli costruiti in CAD e sono memorizzati come singoli file con estensione OFF, ovvero un file contenente una lista di punti e successivamente una lista di facce triangolari i cui vertici provengono dalla prima lista menzionata. Le dieci categorie presenti sono: vasca da bagno, letto, sedia, scrivania, armadio, monitor, comodino, sofa, tavolo e toilette.

Preparazione dei dati

Upsampling / Downsampling

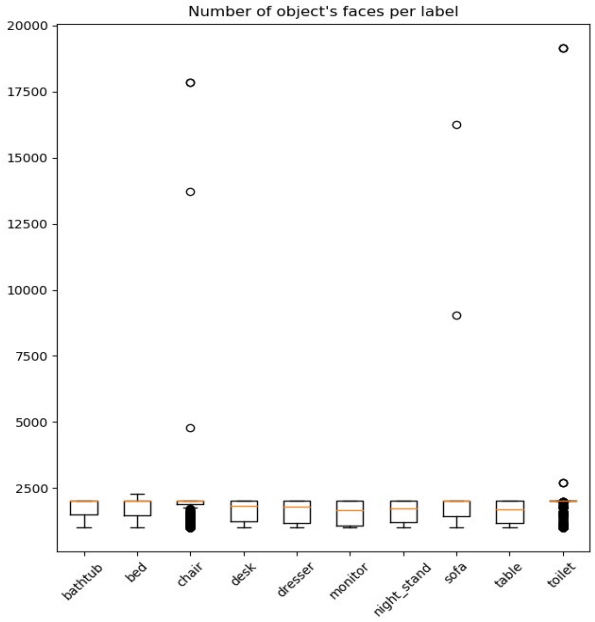

Figura 6: Numero di facce presenti in ogni oggetto, elencati per categoria

Ogni oggetto presente nel dataset contiene un numero variabile di punti e facce, come si può vedere dalla figura 6. Prima di essere usati come input per la rete neurale è stato necessario portare gli oggetti ad un numero arbitrario di punti, in modo tale da costruire dei batch da dare in pasto al modello. Per ottenere ciò ho fissato il numero di punti per oggetto a 2000 e applicato delle tecniche di downsampling e upsampling in base al numero di punti presenti nell'oggetto originario. Il numero arbitrario 2000 è stato scelto considerando sia la memoria disponibile nella GPU sia con un criterio

visivo, notando che gli oggetti con molti punti risultavano ugualmente riconoscibili anche dopo il downsampling. Sfortunatamente, l'operazione di downsampling introduce una perdita di dettagli ed informazione sull'oggetto originale.

Le coordinate degli oggetti sono state successivamente scalate in un cubo di lato unitario con l'incrocio delle sue diagonali centrato nell'origine. Questa operazione distrugge il sistema metrico degli oggetti, impedendo alla rete di classificarli basandosi solo sulle dimensioni. Ad esempio, a parità di sistema metrico adottato, un comodino sarà sempre più piccolo rispetto ad una vasca da bagno o un armadio.

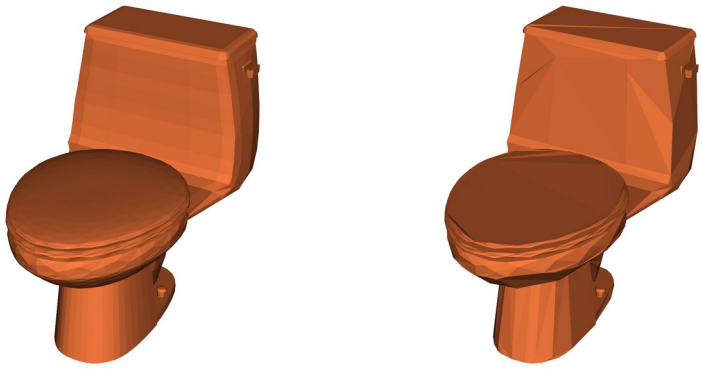

Figura 7: Oggetto proveniente dal dataset prima e dopo l'operazione di downsampling

Class imbalance

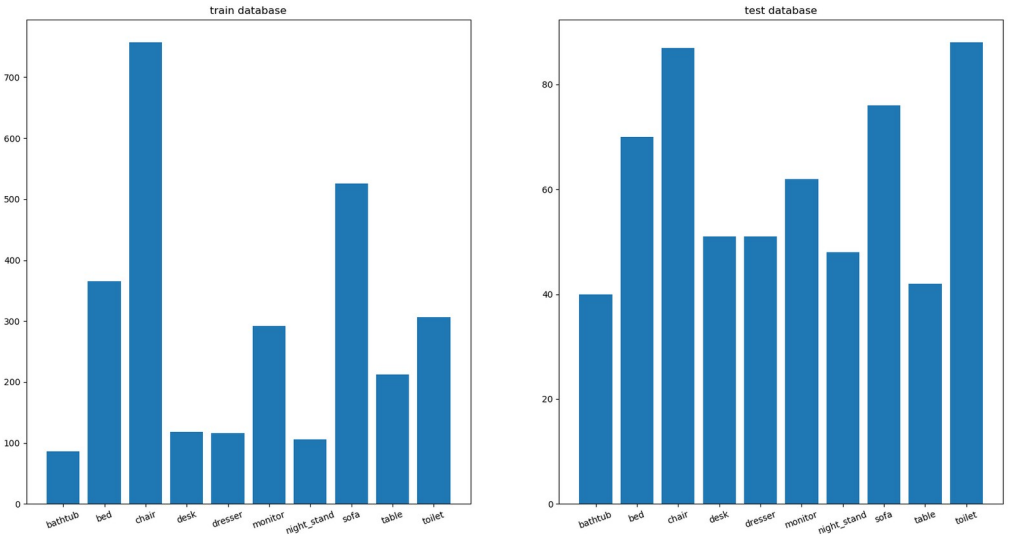

Figura 8: Numero di oggetti per singola classe nel dataset di train ed in quello di test

Un altro problema relativo al dataset è la non conformità nel numero di oggetti per ogni classe. Come si può vedere dal grafico, le classi più rappresentate sono quelle della sedia e del sofa. Lo sbilanciamento tra le varie classi comporta una minore capacità di apprendimento nelle classi meno rappresentate, inficiando poi

sull'accuratezza totale della rete neurale. Solitamente questo problema lo si affronta in due modi diversi: duplicando più volte elementi delle classi meno rappresentate o passando dei pesi per ogni classe all'ottimizzatore. L'ultimo approccio ha il vantaggio di non aumentare i tempi di training del modello e come risultati è comparabile al primo, perciò è stato scelto per questo progetto. I pesi per le varie classi sono stati assegnati nella maniera seguente

$$w_k = \frac{n° \, oggetti \, totali}{(n° \, classi) \cdot (n° \, oggetti \, classe \, k)}$$

Una volta calcolati i pesi vengono integrati nella funzione di loss in modo tale da dare più importanza alle classificazioni errate nelle classi meno rappresentate. Penalizzando maggiormente gli errori nelle classi che la rete ha meno occasione di incontrare nel dataset di training si cerca di bilanciare la diversa rappresentazione delle classi presente.

Approccio adottato

RandLANet

Data una nuvola di punti di grandi dimensioni, per essere processata è necessario effettuare il downsampling ad ogni layer del modello, ovvero estrarre in maniera efficiente alcuni punti da analizzare e da cui la rete possa in maniera efficace ricavarne delle proprietà utili per la classificazione [5]. Per fare ciò viene adottato il metodo di estrazione casuale dei punti per diminuire la densità e la dimensione dei dati da analizzare, senza influire sulla performance dato che, rispetto ad altri metodi di estrazione, esso non dipende dalla dimensione dei dati in input ed ha complessità temporale $O(1)$. Questo approccio, che rispetto ad altri metodi di estrazione è il meno costoso in termini di tempi di esecuzione, presenta alcuni possibili problemi: durante l'estrazione randomica è probabile che alcune parti della point cloud, magari quelle più caratteristiche e distintive, vengano scartate. Per ovviare questo problema viene adottato come blocchetto costituente della rete lo stesso proposto nel paper "RandLANet" in grado di aggregare le proprietà locali della nuvola di punti per poi passarle ai layer successivi per un'ulteriore analisi.

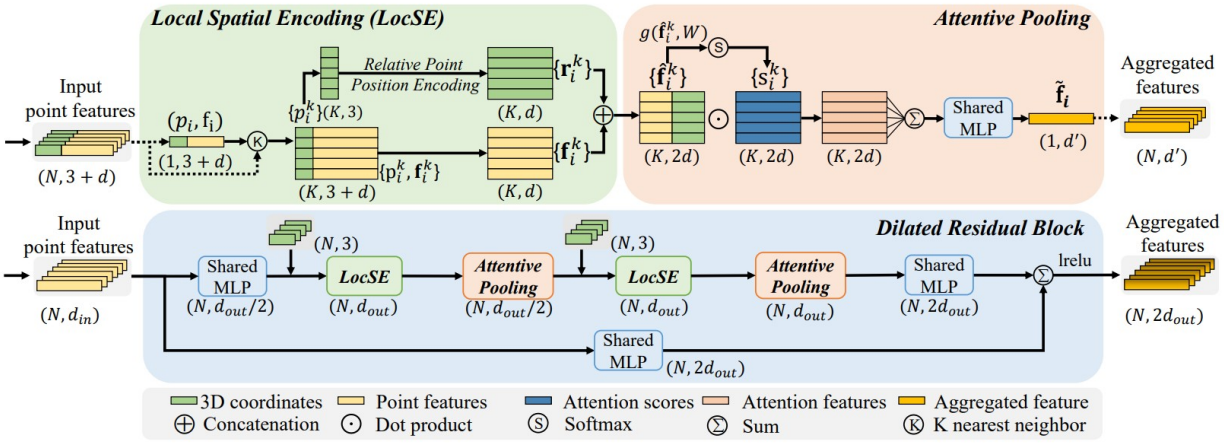

Figura 9: Schema dei moduli che compongono la RandLANet

Local Feature Aggregation

Come mostrato in figura 9, il blocchetto costituente chiamato Local Feature Aggregation è applicato ad ogni punto 3D in parallelo ed è costituito da tre sottounità fondamentali: 1) Local Spatial Encoding, 2) Attentive Pooling e 3) Dilated Residual Block.

Local Spatial Encoding

Data una nuvola di punti **P** con eventualmente degli attributi per ogni punto (come il colore espresso in RGB), il modulo Local Spatial Encoding raggruppa, analizza ed integra le informazioni provenienti dai punti vicini a quelli estratti con l'estrazione randomica, in modo tale che gli attributi associati ad ogni punto estratto contengano la posizione relativa nella zona di spazio in cui si trova codificata. Questo permette al modulo di osservare e cogliere l'andamento locale della geometria della nuvola di punti, fornendo alla rete un modo per rappresentare ed imparare complesse strutture geometriche aggregando le informazioni locali.

Dato il punto i-esimo, i suoi punti vicini sono estratti grazie all'algoritmo kNN basato sulla distanza euclidea tra punti. Per ogni gruppo di punti $\{p_i^1, p_i^2, \ldots, p_i^K\}$ centrati attorno a p_i, la posizione relativa viene codificata secondo la seguente formula:

$$r_i^K = MLP\left(p_i \oplus p_i^k \oplus \left(p_i - p_i^k\right) \oplus \|p_i - p_i^k\|\right)$$

Le informazioni vengono estratte da un vettore ottenuto concatenando i vari punti con la loro differenza con il punto di partenza e con la loro distanza in modulo dal punto di partenza. Questo sembra introdurre informazioni ridondanti nella rete, ma gli esperimenti empirici dimostrano che queste informazioni aiutano la rete a classificare meglio le nuvole di punti.

Le posizioni relative sono poi concatenate agli attributi dei singoli punti, il che codifica esplicitamente le informazioni provenienti da una singola regione locale della nuvola di punti con fulcro nel punto p_i estratto. Questo output è poi passato direttamente al seguente modulo, in grado di aggregare l'insieme di attributi appreso proveniente dai punti vicini.

Attentive Pooling

Il compito del sottoblocco della rete Attentive Pooling è quello di concentrare l'attenzione su alcuni aspetti delle geometrie locali, possibilmente i più utili alla classificazione ed aggregare le informazioni provenienti dai punti in un unico vettore di attributi. Lavori simili di solito utilizzano il metodo max o mean pooling per integrare gli attributi locali, risultandone una perdita di informazioni. L'approccio adottato consiste nell'utilizzare un meccanismo di attenzione per fare in modo che la rete selezioni autonomamente gli attributi locali importanti. Questo attraverso una funzione *g()* imparabile, comune a tutti i punti, che assegna un punteggio ad ogni singolo attributo in base alla sua importanza. La funzione è composta da un MLP (Multi Layer Perceptron) condiviso seguito da un softmax in grado di dare in output valori compresi tra zero ed uno che rappresentano il grado di importanza di ogni singolo attributo. Tali

punteggi sono usati poi nella parte di aggregazione in una somma pesata di tutti gli attributi.

In sintesi, data una nuvola di punti P ed un su punto i-esimo p_i, I moduli della rete appena descritti imparano una rappresentazione della geometria locale e lasciano passare verso l'output solo le informazioni rilevanti utili per la classificazione.

Dilated Residual Block

Dato che le nuove di punti vengono susseguentemente ridotte grazie all'estrazione randomica, è auspicabile aumentare il campo percettivo della rete per ogni singolo punto, in modo tale che i dettagli locali vengano preservati ed arrivino ai layer successivi del modello, anche se alcuni punti vengono scartati. Per ottenere ciò, vengono giustapposti vari moduli di Local Spatial Encoding e Attention Pooling con delle connessioni di bypass, tutto ispirandosi ai lavori sulle dilated convolutions.

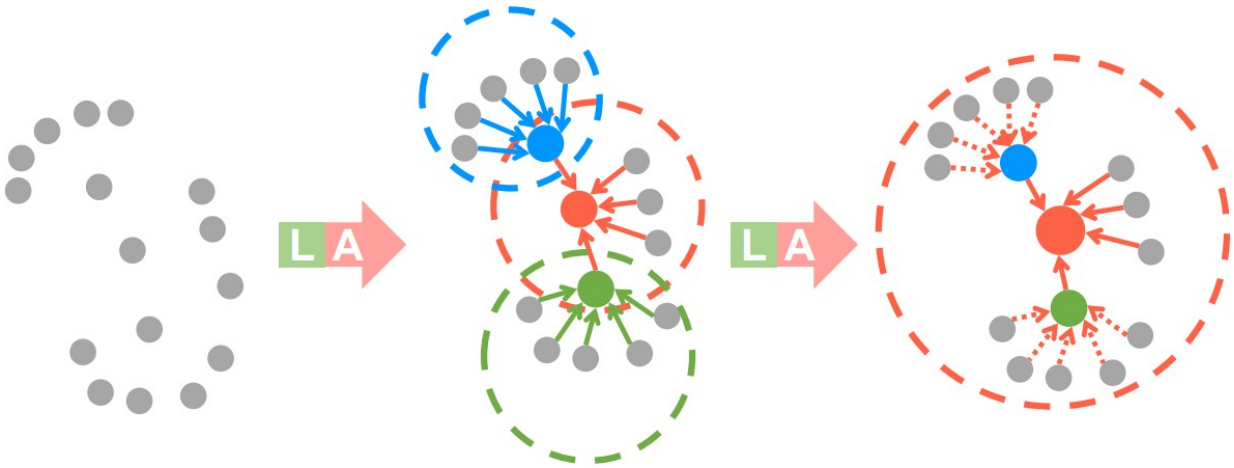

Figura 10: Aumento dei limiti della capacità di elaborazione delle informazioni ad ogni applicazione di Local Spatial Encoding e Attention Pooling

Nella figura 10 si vede come dalla singola nuvola di punti vengono estratti alcuni punti a caso e la rete è in grado, attraverso l'applicazione di vari moduli di Local Spatial Encoding e Attention Pooling di aumentare la sua visione con cui analizzare e processare le informazioni. Inoltre è composta di soli MLP, così da essere computazionalmente veloce e riuscire a scalare la propria complessità in maniera efficiente.

Modello adottato

Nel presente progetto il modello adottato è composto da più layer di Dilated Residual Block, terminanti con dei layer feed-forward classici e un softmax per la classificazione. Il modello è stato scritto utilizzando PyTorch, framework gratuito sviluppato e mantenuto da Facebook.

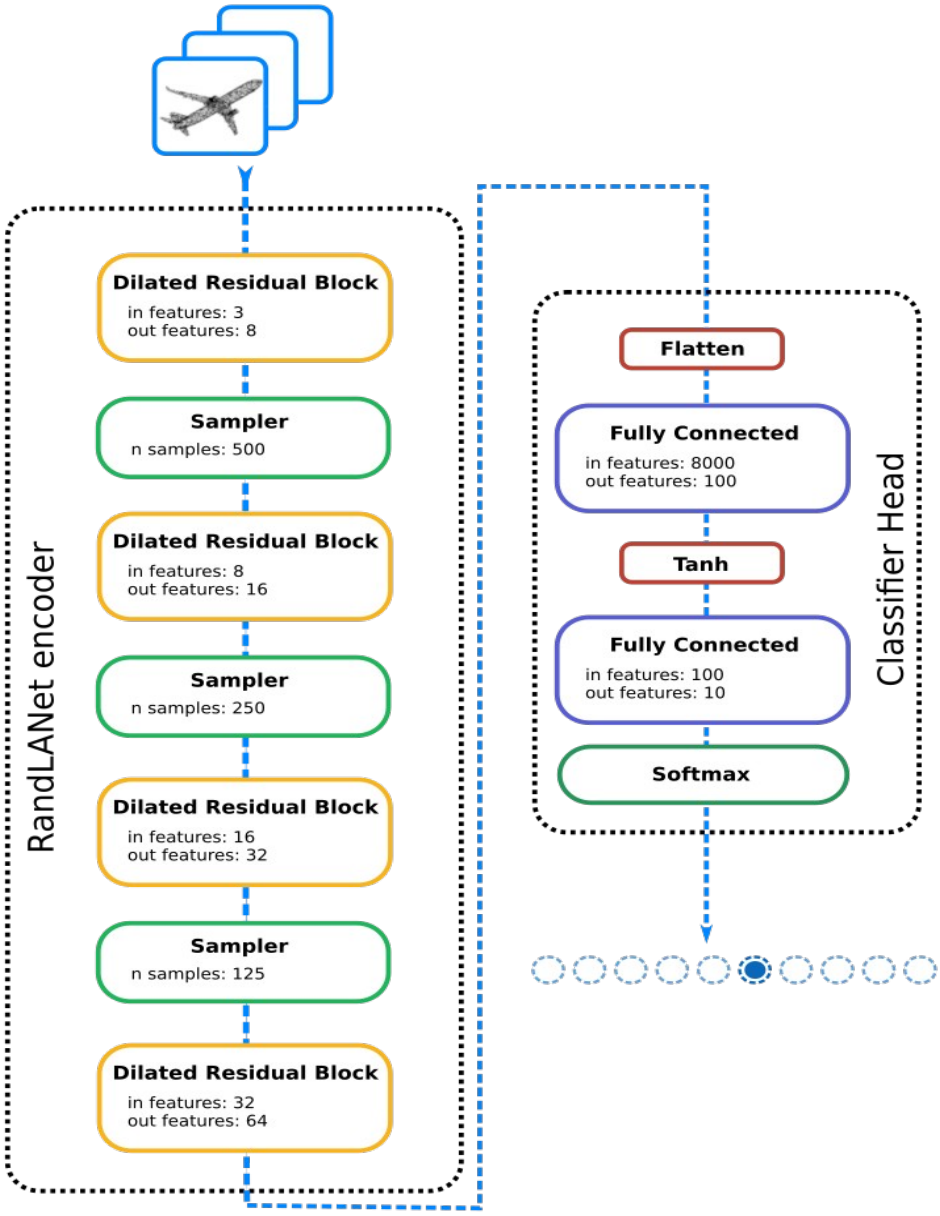

Figura 11: Composizione dei layer della rete neurale

Training

Nella fase di allenamento del modello la ricerca degli iper-parametri è fondamentale per ottenere buone prestazioni dalla rete. Gli iper-parametri in questione sono il learning rate, la dimensione dei batch per l'apprendimento ed il numero di punti presenti in ogni oggetto. La ricerca è stata effettuata grazie ad hyperopt, framework che implementa un algoritmo di ricerca bayesiana capace di trattare spazi di ricerca nel continuo e nel discreto.

Il training è stato effettuato su una scheda grafica nVidia P4000 con 8 Gb di vram.

Figura 12: Risultato migliore del processo di training

Durante l'allenamento del modello utilizzando il dataset di train è stata tenuta traccia dell'andamento della loss function e dell'accuratezza anche nel dataset di test. Dalla figura 12 la funzione di costo nel dataset di test aumenta sebbene aumenti anche l'accuratezza, quando solitamente le due sono inversamente proporzionali. Questo sta a significare che la rete è in grado di classificare correttamente sempre più oggetti, a costo di commettere meno errori ma sempre più grandi, ovvero la distanza tra i vettori $\dot{\mathbf{t}}$ e $\mathbf{t}$ aumenta in maniera sproporzionale.

L’utilizzo del framework hyperopt ha permesso di eseguire diverse iterazioni del processo di training ognuna delle quali con dei parametri diversi alla ricerca della combinazione migliore in grado di ottenere una buona accuratezza. In figura 13 vengono riportati alcuni risultati delle diverse iterazioni, dove il modello viene inizializzato in maniera randomica e allenato sul dataset di train.

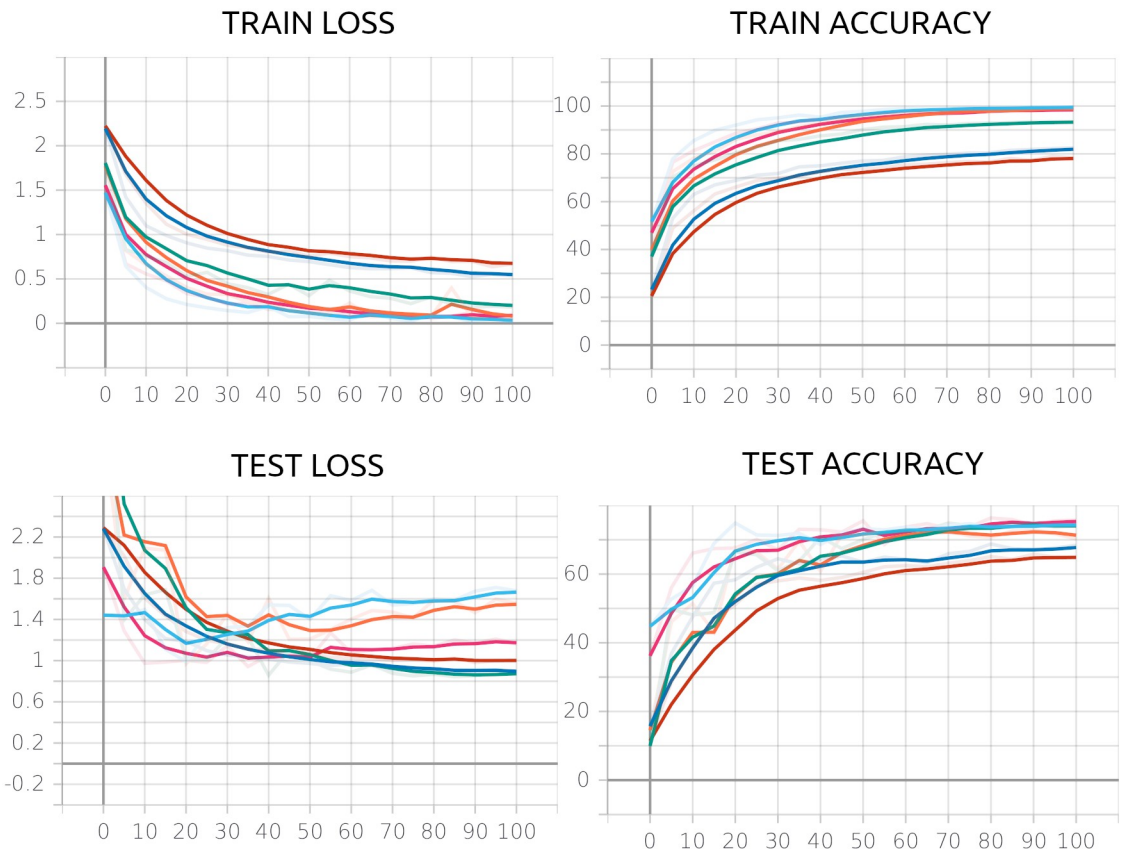

Figura 13: Risultati delle diverse iterazioni del processo di training

Risultati

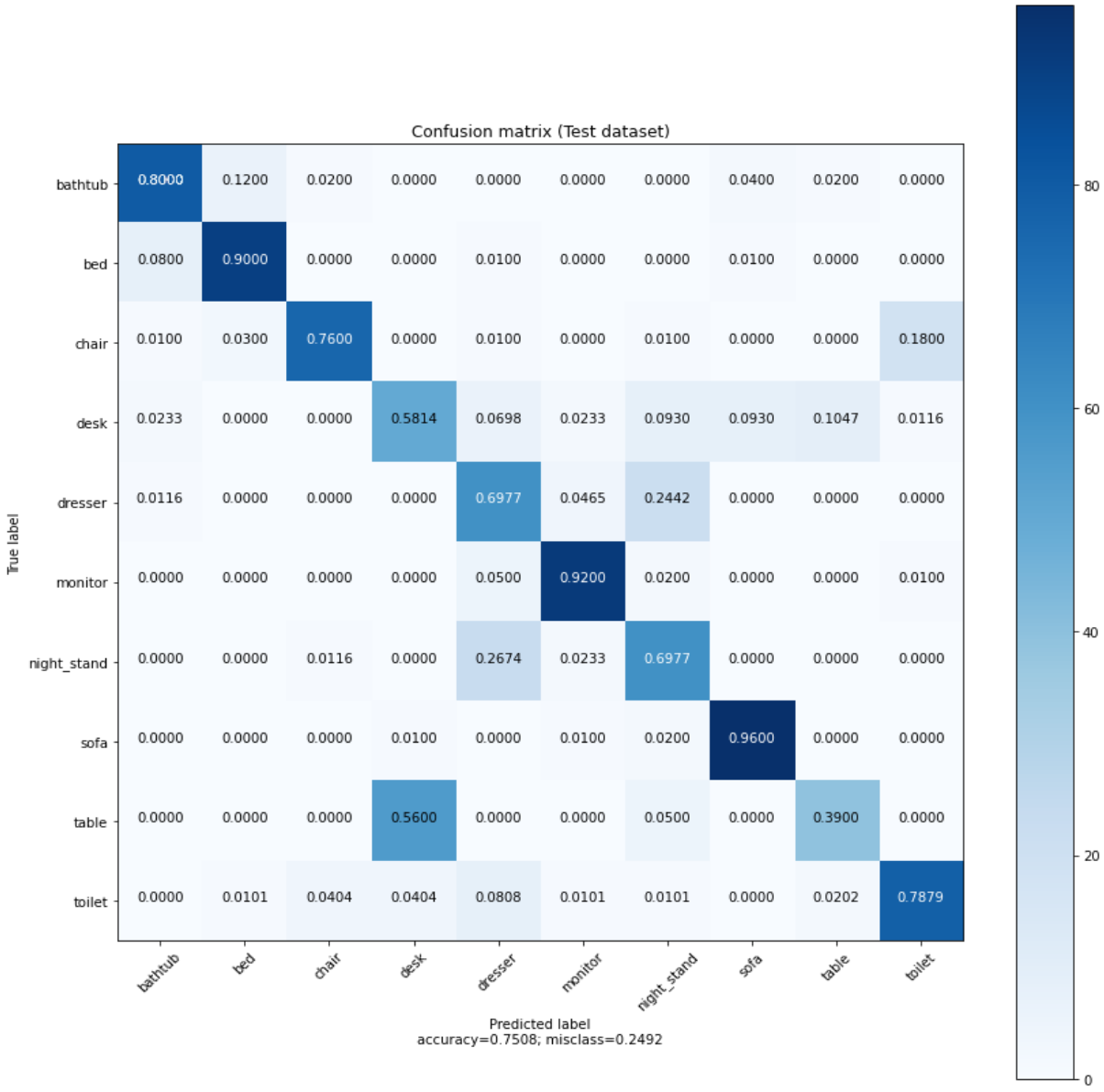

Figura 14: Confusion Matrix

Il modello creato è in grado di raggiungere un'accuratezza del 75% nelle predizioni sul dataset di test, performance leggermente al di sotto dello stato dell'arte. Come si può vedere in figura 12 la rete soffre di overfitting, ovvero raggiunge un'accuratezza vicina al 100% nel dataset con cui è stata allenata ma questo non si traduce poi in un migliore risultato nel dataset di test.

Dalla Confusion Matrix in figura 14 si può vedere che la maggior parte delle classi viene classificata con correttezza, ad eccezione dei tavoli e delle scrivanie che vengono spesso confusi.

Predizioni errate

Data la notevole somiglianza tra alcune classi di oggetti presenti nel dataset, ad esempio i tavoli e le scrivanie, la rete fa fatica a costruirsi delle features che le permettano di classificare correttamente queste tipologie di oggetti. Di seguito in figura 15 vengono riportati alcuni esempi di classificazioni errate presi dal dataset di test.

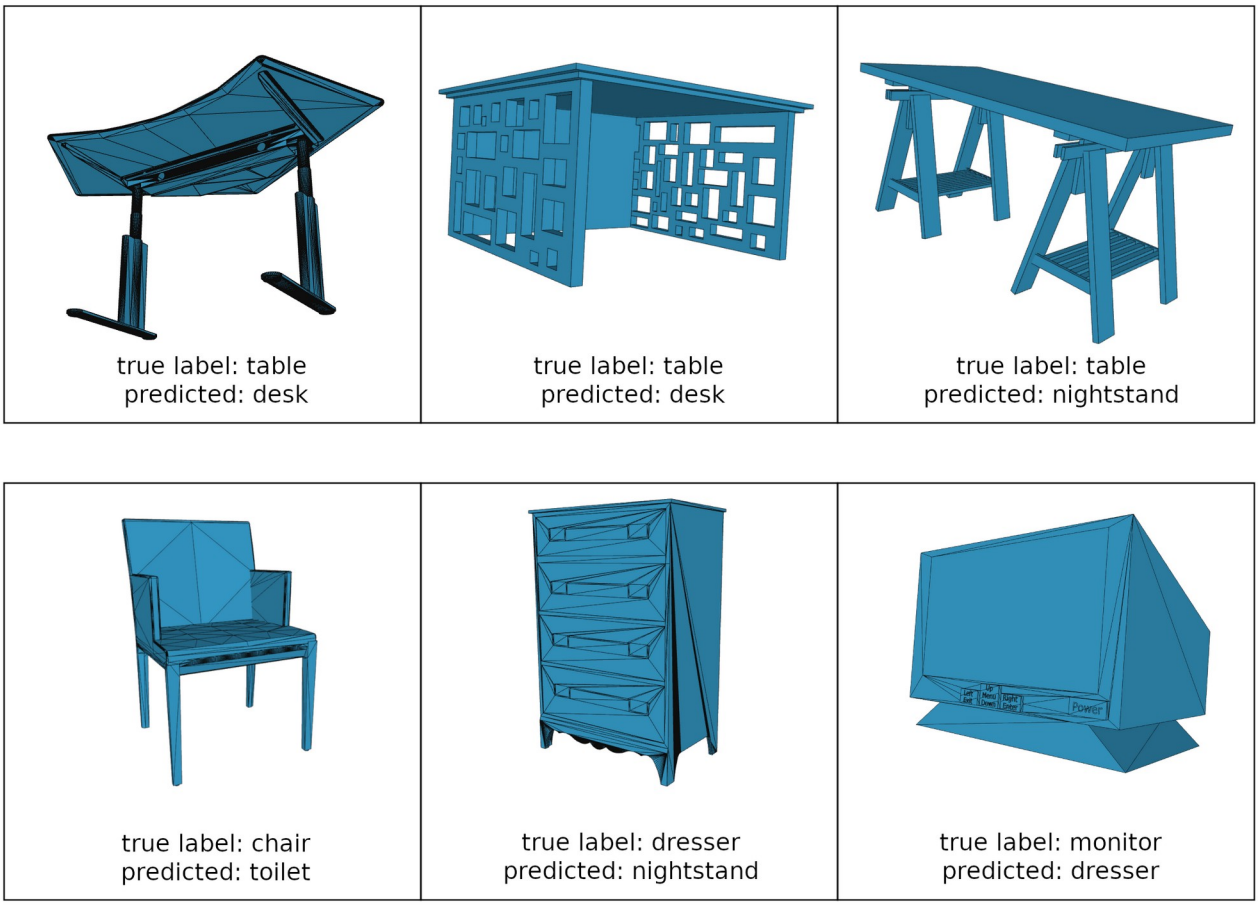

Figura 15: Oggetti reali vs oggetti predetti nel dataset di test

Un elemento ricorrente nelle classificazioni errate è la presenza nell'oggetto di numerosi cluster di punti adibiti a rappresentare parti elaborate dell'oggetto. Dato che i modelli sono stati generati utilizzando il CAD e non sono scansioni di oggetti reali, se sono presenti parti dalle forme complicate esse tendono ad avere più punti rispetto ad altre parti più piane e semplici. Questa irregolarità è una delle sfide che accompagnano le point clouds e che, anche adottando sistemi capaci di aggregare informazioni da punti limitrofi, rimane fonte di errore nel modello creato.

Confronto con lo stato dell'arte

Tipologia Modello	Nome Modello	Accuratezza nel ModelNet10
Multiview	Zanuttigh and Minto	91.5%
	GIFT	92.35%
Voxel	BinVoxNetPlus	92.32%
	VoxNet	92.00%
Graph	Klokov and Lempitsky	94.00%
	ECC	90.00%
Raw point cloud	PointNet	77.6%
	PointNet++	91.9%
	Modello corrente	**74.94%**

In tabella sono riportati i risultati di alcune dei migliori modelli suddivisi per tipologia di approccio adottato per l'analisi delle point cloud. Si può vedere come i metodi basati sull'analisi della point cloud grezza risultino meno performanti. Essi infatti rinunciano a pre-processare i dati in maniera da conformarli ad una struttura più regolare e dalle proprietà sicuramente più vantaggiose, a discapito di una minore performance nel momento dell'esecuzione.

Negli ultimi anni gli approcci basati sull'analisi delle point cloud grezze (ad esempio la PointNet++) hanno raggiunto performance comparabili ai modelli dello stato dell'arte, dimostrando la capacità delle reti neurali di analizzare tipologie di dati fortemente irregolari e in cui le informazioni sono principalmente contenute nelle relazioni tra gli elementi costituenti, nel presente caso nelle distanze relative tra i punti costituenti la point cloud.

Bibliografia

Bibliography

1: Saifullahi Aminu Bello et. al., Review: Deep Learning on 3D Point Clouds, 28 Maggio 2020

2: Pietro Zanuttigh, Ludovico Minto, DEEP LEARNING FOR 3D SHAPE CLASSIFICATION FROM MULTIPLE DEPTH MAPS, 2017

3: Christopher M. Bishop, Pattern Recognition and Machine Learning, 2006

4: Princeton, ModelNet, , https://modelnet.cs.princeton.edu/

5: Qingyong Hu et. al., RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds,