



**Politecnico
di Torino**

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria del Cinema e dei Mezzi di
Comunicazione

Ottobre 2023

Analisi delle prestazioni di reti neurali per la ricostruzione di oggetti 3D

Relatori:

Andrea Sanna

Federico Manuri

Candidato:

Emanuela Favasuli

Sommario

L'avanzamento delle capacità di calcolo dei computer e l'accessibilità sempre più ampia dalle tecnologie di apprendimento automatico nell'ambito dell'intelligenza artificiale e della realtà aumentata hanno un impatto significativo al giorno d'oggi. In particolare, la disponibilità di librerie deep learning e il potenziamento dell'hardware hanno reso possibile l'addestramento di reti neurali per digitalizzare oggetti reali in ambienti virtuali. Inizialmente, le reti neurali erano principalmente utilizzate per compiti come il riconoscimento vocale e la classificazione di oggetti, ma con l'avanzare dello sviluppo di algoritmi sempre più sofisticati, oggi sono in grado di ricostruire modelli tridimensionali partendo da immagini 2D.

La mancanza di standardizzazione nei dataset per l'addestramento delle reti neurali rende complesso comprendere quali parametri delle immagini siano più influenti sulla qualità di ricostruzione 3D. Tale sfida è stata affrontata nella tesi intitolata "Creazione di dataset di immagini sintetiche per l'addestramento di reti neurali dedicate alla generazione automatica di oggetti 3D", condotta dall'ing. Ismaele Piparo, il cui scopo è stato quello di esplorare quale fosse l'impatto dei vari parametri delle immagini sintetiche, usate per il training di una rete neurale, sulla qualità dell'oggetto 3D ricostruito. Quest'ultimo elaborato, è stato utilizzato come punto di riferimento nel presente lavoro per valutare l'adeguatezza dell'approccio metodologico delineato dall'ingegnere per la rete neurale BSP-Net, estendendolo ad altre reti neurali con architetture diverse da ResNet.

Inizialmente, è stata condotta un'analisi dettagliata delle diverse reti neurali coinvolte nella ricostruzione 3D, concentrandosi in particolare su quelle in grado di eseguire una ricostruzione da singola vista partendo da immagini RGB. Durante la fase di analisi, diverse reti neurali convoluzionali sono state esaminate con l'obiettivo specifico di allontanarsi dall'architettura di ResNet-18 e sue varianti. La selezione è stata basata su criteri precisi, includendo la necessità che le reti neurali non richiedessero un addestramento su dataset esterni, che fossero in grado di operare con immagini sintetiche e fossero compatibili con le classi di oggetti di ShapeNet. Inoltre, era fondamentale che queste reti non fossero vincolate all'utilizzo di sfondi chiari nelle immagini di input. Dopo una ricerca approfondita, la rete neurale selezionata come caso studio per questo lavoro è stata *Pix2Vox*. *Pix2Vox* è una rete neurale sviluppata in Python, implementata in Pytorch, che permette la ricostruzione di oggetti 3D in formato voxel a partire da una singola immagine in input (Single-view Reconstruction). In particolare, *Pix2Vox* lavora utilizzando come input immagini con dimensione 224x224px.

Per l'implementazione della rete, sono stati installati i requisiti specifici forniti dall'autore e sono stati configurati accuratamente i parametri. Solo dopo aver analizzato quali fossero i parametri cruciali per il buon funzionamento della rete, si è passati ad una fase di ottimizzazione in cui sono state apportate delle modifiche per migliorare le prestazioni della rete. Tuttavia, tali modifiche sono state effettuate in base alla capacità del sistema e alla quantità di dati disponibili per evitare un sovraccarico.

In una fase successiva, la rete neurale selezionata è stata configurata in maniera tale che lavorasse con lo stesso dataset di immagini usato nella tesi di riferimento menzionata in precedenza.

Questo dataset è stato generato raccogliendo modelli tridimensionali rappresentativi di diverse categorie di oggetti ed effettuando diverse fasi di rendering, tramite il software Blender, per ottenere le relative immagini sintetiche. Poiché era già emerso che vari fattori, come la risoluzione delle immagini, le condizioni di illuminazione, la presenza o l'assenza di materiali, la posizione della camera, influenzano il processo di addestramento della rete, si è scelto di organizzare il dataset in categorie corrispondenti a specifiche tipologie di oggetti, ognuna delle quali presenta caratteristiche specifiche.

Tra le diverse categorie di oggetti, per l'addestramento di Pix2Vox, la categoria selezionata è stata quella delle sedie. Prima di procedere con l'addestramento, essendo il modello già pre-addestrato col dataset ShapeNet, è stato necessario creare un file di tassonomia specifico per ogni diversa categoria delle sedie. Tale file conteneva l'identificativo della categoria di oggetti, il nome della categoria e la suddivisione dei vari modelli in fasi di addestramento, test e convalida.

Successivamente, nel contesto della configurazione della rete, si è passati alla gestione dell'accesso ai dati del dataset di riferimento, il quale è stato denominato "*Piparo*".

L'addestramento in fase iniziale è stato eseguito ridimensionando le immagini del dataset alle dimensioni operative della rete Pix2Vox, ovvero 224x224px. In seguito, la rete è stata sottoposta ad ulteriori cicli di addestramento utilizzando immagini di dimensioni minori.

Per la fase di training della rete neurale, sono state impiegate immagini sintetiche, mentre per la fase di ricostruzione sono state utilizzate immagini "reali" provenienti dal dataset Pix3D. Questo dataset ha consentito l'utilizzo delle immagini reali per la ricostruzione degli oggetti, mentre i modelli 3D sono stati utilizzati come "*ground truth*" per valutare la qualità della ricostruzione. Per analizzare la fedeltà dell'oggetto ricostruito rispetto al modello di riferimento, è stata adottata una metrica di valutazione nota come *Chamfer Distance*. Questa

metrica è stata progettata per misurare la somiglianza tra i modelli 3D ricostruiti e quelli di riferimento, entrambi rappresentati come nuvole di punti.

Poiché la rete Pix2Vox non possiede la capacità intrinseca di generare direttamente nuvole di punti, è stato adottato un approccio indiretto integrando nel codice uno script Python precedentemente sviluppato, per calcolare la Chamfer Distance.

La fase di training della rete neurale e la fase successiva di ricostruzione dei modelli 3D sono stati ripetuti per ciascuna delle categorie, in base ai diversi attributi di cui si volevano valutare gli effetti.

Infine, dopo aver condotto una serie di esperimenti, è stato eseguito un confronto tra i risultati ottenuti nelle diverse prove e quelli ottenuti negli esperimenti riportati nella tesi precedente. Tuttavia, l'intenzione principale di questo confronto non è stata quella di esaminare i valori assoluti della metrica di valutazione, bensì quello di analizzare l'andamento di quest'ultima al variare dei parametri fondamentali che influenzano il processo di addestramento e la successiva qualità di ricostruzione 3D.

Questo tipo di analisi ha permesso di confrontare l'efficacia e la robustezza di due reti neurali considerate in diverse condizioni e comprendere come i diversi parametri, quali risoluzione delle immagini, condizioni di illuminazione, presenza o meno di materiali, posizione della camera, impattino sulla capacità della rete neurale di apprendere e ricostruire in modo accurato gli oggetti 3D, consentendo così di identificare le condizioni ottimali per ottenere risultati migliori nelle applicazioni di generazione automatica di oggetti tridimensionali.

Indice

Elenco delle figure	6
Elenco delle tabelle	6
1 Introduzione	11
1.1 Contesto	11
1.2 Descrizione dell'elaborato precedente	12
1.2.1 BSP-Net	12
1.2.2 Descrizione del lavoro	20
1.2.3 Risultati ottenuti	24
1.3 Obiettivo della tesi	27
2 Stato dell'arte	30
2.1 Reti neurali per la ricostruzione di elementi 3D	30
2.2 Principali dataset di immagini	32
2.3 Altri studi nella generazione di dataset	36
3 Tecnologie utilizzate	38
3.1 Reti convoluzionali	38
3.1.1 CoreNet	39
3.1.2 DISN	40
3.1.3 D ² IM-Net	41
3.1.4 Image2Mesh	42
3.1.5 Sym3DNet	44
3.2 Pix2Vox	45
3.2.1 Fase di training	46
3.2.2 Ricostruzione degli oggetti 3D	49
3.3 Pix3D	51
3.4 Linguaggio Python	52
4 Pipeline di lavoro e metriche di valutazione	55
4.1 Descrizione pipeline per la creazione del dataset	55
4.2 Feature per il training	56
4.2.1 Dimensione dell'immagine	56
4.2.2 Texture e materiali	57
4.2.3 Illuminazione	58
4.3 Chamfer Distance	59
4.4 Altre metriche di valutazione	61
5 Progettazione e realizzazione del lavoro	63

5.1	Rete Neurale	63
5.1.1	Implementazione della rete neurale	63
5.2	Addestramento della rete	66
5.2.1	Tassonomia	67
5.2.2	Lettura file	68
5.2.3	Ridimensionamento immagini	70
5.3	Ricostruzione 3D e calcolo della Chamfer Distance	70
5.3.1	Voxel	71
5.3.2	Calcolo Chamfer Distance	72
6	Test e Risultati ottenuti	73
6.1	Selezione dei modelli	73
6.2	Esperimenti	74
6.2.1	Dimensione delle immagini	74
6.2.2	Distanza della camera	75
6.2.3	Materiali e texture	75
6.2.4	Illuminazione	77
6.3	Risultati ottenuti	78
6.3.1	Dimensione dell'immagine	78
6.3.2	Distanza della camera e presenza di materiali	81
6.3.3	Materiali e texture	84
6.3.4	Illuminazione	87
6.4	Confronto grafico	88
7	Conclusioni e sviluppi futuri	92
7.1	Conclusioni	92
7.2	Eventuali miglioramenti e sviluppi futuri	94
A	Requisiti della workstation e versione dei software	96
A.1	Workstation	96
A.2	Software	96
	Riferimenti bibliografici	97

Elenco delle figure

1.1	Rappresentazione dei principali livelli di una rete CNN.	13
1.2	Immagine di input che viene passata attraverso uno strato convoluzionale.	14
1.3	Esempio di operazione di convoluzione.	14
1.4	Il livello pooling riduce le dimensioni dei parametri tramite le funzioni max pooling e average pooling.	15
1.5	Processo di training della rete BSP-Net.	17
1.6	Processo di ricostruzione 3D a partire da un'immagine.	18
1.7	Principali moduli della pipeline di BlenderProc.	19
2.1	Rappresentazione di due tecniche comuni.	31
2.2	Diverse rappresentazioni di un modello 3D.	32
2.3	Partendo da un'immagine a colori naturali, la colonna a sinistra mostra i canali di colore isolati RGB, mentre a destra ci sono le loro equivalenze in scala di grigi.	33
2.4	Alcune tipologie di modelli 3D pronti all'uso.	35
3.1	Architettura base di una rete CNN.	38
3.2	Architettura della rete CoReNet. Il dato viene propagato dall'encoder al decoder tramite le connessioni di skip-ray traced (freccie rosse).	40
3.3	Funzionamento della rete DISN.	40
3.4	Struttura della rete D ² IM-Net composta da tre fasi.	42
3.5	Architettura della rete neurale Image2Mesh.	43
3.6	Funzionamento della rete Sym3DNet.	44
3.7	Una visione della rete proposta Pix2Vox.	45
3.8	Architettura della rete Pix2Vox-F (alto) e Pix2Vox-A (sotto).	46
3.9	Illustrazione funzionamento delle mappe di punteggio.	48
3.10	A partire dall'immagine in input, ricostruzione in formato voxel rispetto al modello di riferimento.	51
3.11	Esempi di immagini e forme in Pix3D. Da sinistra a destra: forme 3D, immagini 2D e allineamento immagine-forma.	52
4.1	Percezione dei dettagli in base alla diversa risoluzione dell'immagine.	57
4.2	Differenti mappe di texture.	58
4.3	Sistema three-point lighting.	59
4.4	Rappresentazione grafica della Chamfer Distance.	60
4.5	Formula matematica per la Chamfer Distance.	60
4.6	Formula matematica della metrica IoU.	61
4.7	Rappresentazione del compito di rilevamento dell'oggetto.	62

5.1	Suddivisione del dataset.	68
5.2	Modello in formato voxel generato col tool Binvov.	71
6.1	Rappresentazione di un'immagine sintetica di una sedia a risoluzione 128x128px e 224x224px.	74
6.2	Rappresentazione di un modello con distanza fissa (a sinistra) e variazione della camera (a destra).	75
6.3	Immagine di uno stesso modello senza materiale (a sinistra) e con applicazione del materiale (a destra).	76
6.4	Esempio di un modello con pattern.	77
6.5	Immagini presenti nel (a) primo dataset, (b) secondo dataset e (c) terzo dataset.	79
6.6	Esempio di immagini dal dataset di Pix3D impiegate nel processo di ricostruzione degli oggetti.	83
6.7	Analisi dell'andamento della Chamfer Distance, per la rete Pix2Vox, al variare dei parametri chiave.	89
6.8	Analisi dell'andamento della Chamfer Distance, per la rete BSP-Net, al variare dei parametri chiave.	90

Elenco delle tabelle

1.1	Variazione delle prestazioni di ricostruzione in funzione dei vari parametri.	27
6.1	Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il primo dataset di immagini.	79
6.2	Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il secondo dataset di immagini.	79
6.3	Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il terzo dataset di immagini.	80
6.4	Confronto dei tempi medi di training delle due reti neurali per i tre dataset di immagini.	81
6.5	Valori di Chamfer Distance per Pix2Vox relativi a un dataset di immagini in cui la camera è mantenuta a una distanza fissa dal modello 3D e un dataset in cui la camera varia la sua distanza dal modello.	82
6.6	Valori di Chamfer Distance per BSP-Net relativi a un dataset di immagini in cui la camera è mantenuta a una distanza fissa dal modello 3D e un dataset in cui la camera varia la sua distanza dal modello.	82
6.7	Valori di Chamfer Distance per Pix2Vox relativi a un dataset di immagini ottenuti da modelli con materiale applicato e un dataset in cui i modelli sono senza materiale.	84
6.8	Valori di Chamfer Distance per BSP-Net relativi a un dataset di immagini da modelli con materiale applicato e un dataset in cui i modelli sono senza materiale.	84
6.9	Valori di Chamfer Distance per Pix2Vox relativo ad immagini con materiale “realistico” e immagini con solo la color map.	85
6.10	Valori di Chamfer Distance per BSP-Net relativo ad immagini con materiale “realistico” e immagini con solo la color map.	85
6.11	Valori di Chamfer Distance per Pix2Vox relativi al dataset di immagini aggiungendo le altre mappe ai materiali.	86
6.12	Valori di Chamfer Distance per BSP-Net relativi al dataset di immagini aggiungendo le altre mappe ai materiali.	86
6.13	Confronto dei valori di Chamfer Distance per le due reti relativi al dataset di immagini con specifici pattern.	87
6.14	Valori di Chamfer Distance per Pix2Vox relativi ad esperimenti con modelli illuminati in maniera omogenea e modelli illuminati con la configurazione del three-point lighting.	87

6.15 Valori di Chamfer Distance per BSP-Net relativi ad esperimenti
con modelli illuminati in maniera omogenea e modelli illuminati
con la configurazione del three-point lighting. 88

1 Introduzione

1.1 Contesto

L'avanzamento delle capacità di calcolo dei computer e l'accessibilità a strumenti di apprendimento automatico hanno rivoluzionato il panorama dell'intelligenza artificiale (IA) e della realtà aumentata (AR). Lo sviluppo delle tecnologie hardware e l'uso di processori sempre più veloci, hanno permesso a ricercatori e sviluppatori di esplorare nuovi orizzonti nell'ambito dell'IA e dell'apprendimento automatico (Machine learning¹). Negli ultimi anni, la disponibilità di librerie di deep learning e di molte piattaforme per la creazione e gestione di modelli di IA, ha reso possibile l'utilizzo di potenti macchine virtuali per l'addestramento di reti neurali.

L'utilizzo di reti neurali gioca un ruolo cruciale, in quanto consente agli oggetti reali di essere digitalizzati e inseriti in ambienti virtuali o "metaversi", creando un'unica interazione tra il mondo fisico e quello digitale. Utilizzate nel contesto del deep learning² ed ispirate dal funzionamento dei neuroni del cervello umano, consentono alle macchine di apprendere dai dati ed eseguire attività complesse come il riconoscimento della voce, la classificazione di oggetti, la guida autonoma, la gestione degli NPC (Non Playable Character) e molto altro.

In base al tipo di problema e alla complessità della rete, l'addestramento di queste reti neurali è variabile.

In particolare, le reti neurali specializzate nella ricostruzione di oggetti 3D vengono allenate attraverso dataset di immagini rappresentanti oggetti da diverse angolazioni e pose, permettendo loro di ricostruire fedelmente oggetti reali in ambienti virtuali. Tuttavia, non esistono criteri standard per la creazione di dataset sintetici da utilizzare nell'addestramento. Spesso sono utilizzati dataset di

¹Il *machine learning* è una sottodisciplina dell'intelligenza artificiale (AI) che si occupa dello sviluppo di algoritmi e modelli computazionali, i quali consentono alle macchine di apprendere una molteplicità di dati strutturati ed etichettati per eseguire compiti specifici.

²Il *deep learning* è un ramo del machine learning che si concentra sull'uso di reti neurali profonde, ovvero costituite da molti strati interconnessi, per l'estrazione di rappresentazioni complesse dai dati.

immagini sintetiche generate da modelli 3D tramite rendering.

La diversità tra i vari dataset rende però difficile comprendere quali parametri delle immagini siano più rilevanti per il buon funzionamento di una rete neurale e, di conseguenza, non è possibile stabilire in modo definitivo come tali parametri, quali risoluzione dell'immaginazione, tipo di illuminazione, presenza di texture, possano influenzare le prestazioni della rete stessa. Questa mancanza di standardizzazione non solo rende l'addestramento delle reti di ricostruzione 3D più complesso, ma contribuisce anche ad una continua di ricerca e sperimentazione per individuare le migliori pratiche.

1.2 Descrizione dell'elaborato precedente

1.2.1 BSP-Net

La rete neurale usata come termine di paragone per questo lavoro di tesi è "*BSP-Net: Generating Compact Meshes via Binary Space Partitioning*" [8]. BSP-Net è una rete neurale che permette la ricostruzione di modelli 3D attraverso il "Binary Space Partition" (BSP), un metodo per il partizionamento dello spazio Euclideo, il quale viene suddiviso in due insiemi convessi utilizzando gli iperpiani come partizione. Tale rete, costruita sul modello base delle reti convoluzionali (CNN) [20], permette la ricostruzione *Single-View*, ossia ricostruisce la mesh 3D a partire da una singola immagine di un oggetto. BSP-Net è stata usata nella tesi intitolata "*Creazione di dataset di immagini sintetiche per l'addestramento di reti neurali dedicate alla generazione automatica di oggetti 3D*" [22], condotta dall'ing. Ismaele Piparo, e servirà come riferimento nel presente lavoro per verificare come le reti neurali progettate per la ricostruzione di oggetti 3D si comportano al variare di determinati parametri come la risoluzione dell'immagine di input, la posizione della camera, i materiali degli oggetti, l'illuminazione della scena e così via.

In particolare, le reti convoluzionali (CNN) sono delle reti artificiali che sfruttano delle operazioni di convoluzione tra diversi livelli (layer), ognuno dei quali ha la capacità di estrarre da un'immagine di input una matrice di caratteristiche (feature map).

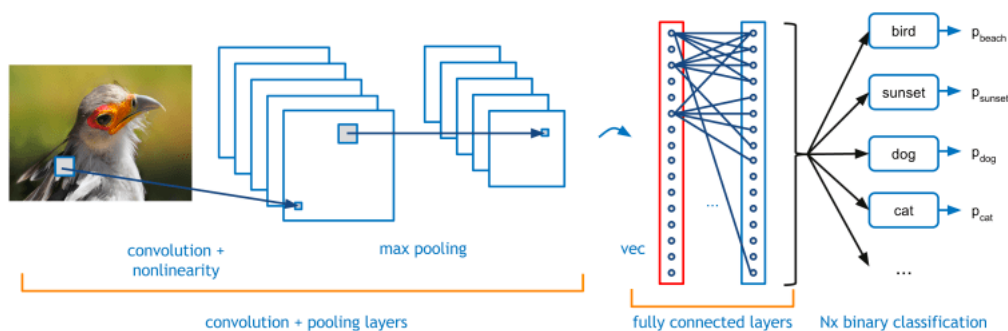


Figura 1.1: Rappresentazione dei principali livelli di una rete CNN.

La denominazione *reti convoluzionali* deriva dalla presenza e dall'importanza degli strati di convoluzione all'interno di queste reti. La convoluzione è un'operazione in cui un filtro, detto *kernel* scorre attraverso l'input e calcola il prodotto scalare tra il filtro e la porzione dell'input che sta esaminando in quel momento. Ripetendo il processo per diverse porzioni sull'input, si genera una mappa delle caratteristiche o *feature map* che cattura le informazioni significative sull'input originale.

Le reti neurali convoluzionali si distinguono dalle altre reti neurali per le loro prestazioni superiori e la loro struttura è costituita da tre livelli principali, come mostrato in figura 1.1:

1. Convolutional Layer (Livello convoluzionale)

Lo strato convoluzionale (figura 1.2), è il componente principale che svolge una serie di calcoli per l'analisi delle immagini. Le operazioni avvengono tramite un filtro *kernel*, ovvero un array 2D di pesi, che viene spostato sull'immagine di input, rappresentata come una matrice 3D, per rilevare specifiche caratteristiche.

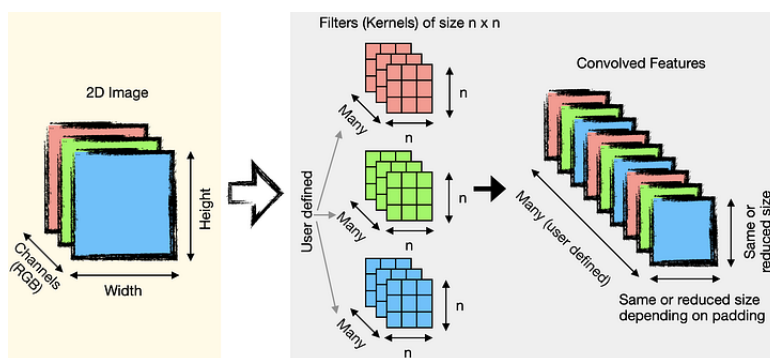


Figura 1.2: Immagine di input che viene passata attraverso uno strato convoluzionale.

La convoluzione viene applicata tra il filtro e l'input, calcolando il prodotto di punti tra i pixel dell'input e i pesi del filtro (figura 1.3). Questo processo si ripete mentre il filtro si sposta sull'intera immagine. Dopo la convoluzione, viene applicata una funzione di attivazione ReLU (Rectified Linear Unit) per introdurre non linearità.

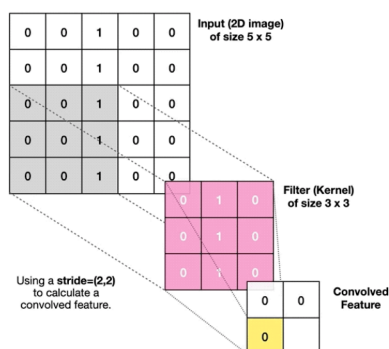


Figura 1.3: Esempio di operazione di convoluzione.

Grazie a questo livello, la rete è in grado di estrarre feature rilevanti consentendo la creazione di gerarchie di pattern per compiti di riconoscimento e classificazione di un oggetto.

2. Pooling layer (Livello di Pooling)

Il livello di pooling, noto anche come *sottocampionamento*, è un componente che riduce le dimensioni della feature map eseguendo una riduzione del numero di parametri in input. Tuttavia, diminuendo le dimensioni dell'immagine (reshape), riesce a mantenere le caratteristiche della stessa. Questa operazione viene effettuata scansionando la feature map tramite una matrice di dimensioni più piccole; man mano che la matrice scorre sulla feature map si estraggono i valori che andranno a generare una nuova feature map di dimensioni minori. Questo corrisponde ad uno scalamento della feature map, prendendo o i valori medi, *average pooling* o il valore massimo, *max pooling*. Le operazioni del livello di pooling sono mostrate in figura 1.4.

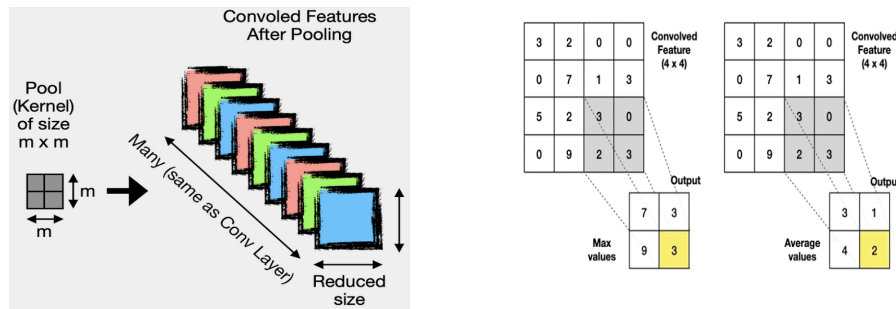


Figura 1.4: Il livello pooling riduce le dimensioni dei parametri tramite le funzioni max pooling e average pooling.

Sebbene il pooling layer comporti la perdita di alcune informazioni dettagliate, è benefico in quanto contribuisce a ridurre la complessità, migliorare l'efficienza computazionale e prevenire il sovradattamento³.

³Il sovradattamento, anche detto *overfitting*, è un problema comune nell'apprendimento automatico che si verifica quando la rete neurale si adatta eccessivamente ai dati di addestramento, non memorizzando bene i nuovi dati non visti. Quando il modello impara "troppo" dai dati di addestramento, inclusi i dettagli e il rumore, può portare a modelli che non sono in grado di generalizzare bene e quindi inutili nella pratica.

3. Fully-connected layer (Strato Completamente Connesso)

Il livello completamente connesso è chiamato così poiché ogni nodo in questo livello è direttamente collegato a ciascun nodo nel livello precedente. Questo livello svolge il compito di classificazione utilizzando le informazioni estratte dai livelli precedenti, inclusi i filtri dei livelli convoluzionali e le operazioni di pooling. Mentre i livelli convoluzionali e di pooling spesso utilizzano la funzione di attivazione ReLU per l'estrazione delle caratteristiche, i livelli completamente connessi applicano una funzione di attivazione *softmax*, la quale assegna una probabilità alle diverse classi di output per determinare la classe più probabile a cui appartiene l'input.

BSP-Net è organizzata secondo lo standard di ResNet-18 [11], una particolare rete convoluzionale. ResNet (Residual Network) sfrutta il concetto di "residuo" per massimizzare le prestazioni, in particolar modo si parla di "blocco residuo". Questo concetto nasce al fatto che le prime reti CNN utilizzavano più livelli convoluzionali possibili per ridurre il tasso di errore, ma non sempre ciò funzionava. Infatti, incrementando sempre più i livelli si rischiava di incrementare il tasso di errore di allenamento e test. Per far fronte a ciò, si è deciso di aggiungere una "scorciatoia" per far fluire maggiormente le informazioni da un livello al livello successivo, senza danneggiare le prestazioni della rete.

La fase di training è fondamentale per il corretto funzionamento di una rete neurale. Ogni rete neurale prevede una fase di addestramento, in cui la rete viene allenata mediante un dataset di immagini rappresentanti diversi oggetti ripresi da diverse angolazioni e pose. Tuttavia, tale fase ha un dispendio variabile sia in termini di durata che in termini di risorse, il quale dipende dalla profondità della rete e dalla dimensione del dataset di immagini. Nello specifico, una sessione di training consiste nel far "esaminare" alla rete neurale l'intero dataset per un numero determinato di volte, detto "*epoca*". Per ogni epoca, vengono analizzati tutti i dati presenti nel dataset, e alla fine di ognuno viene calcolato un errore (training error), ovvero la differenza tra l'output che la rete fornisce rispetto al dato in input. Nel caso di un corretto funzionamento del processo, l'errore tenderà a diminuire col passare delle epoche. Alla fine di ogni sessione di training, ossia dopo aver eseguito un determinato numero di epoche, la rete neurale avrà ottimizzato i vari parametri interni per i vari strati, detti "*pesi*", i quali verranno estratti e salvati. Nello specifico, la rete neurale BSP-Net è costituita da due encoder: *autoencoder*, utilizzato per il training dei parametri relativi alla ricostruzione dei modelli 3D, e *image encoder*, dedicato alla Single

View Reconstruction. Per quest'ultima, il processo di training è suddiviso in due fasi, come mostrato in figura 1.5:

1. Durante la prima fase, training dell'autoencoder, la rete viene allenata tramite modelli rappresentati mediante voxel; sfruttando varie convoluzioni, si allena ad estrarre per ogni voxel delle feature spaziali (predittori), utili per riconoscere sia oggetti simili che per la ricostruzione dei modelli 3D.

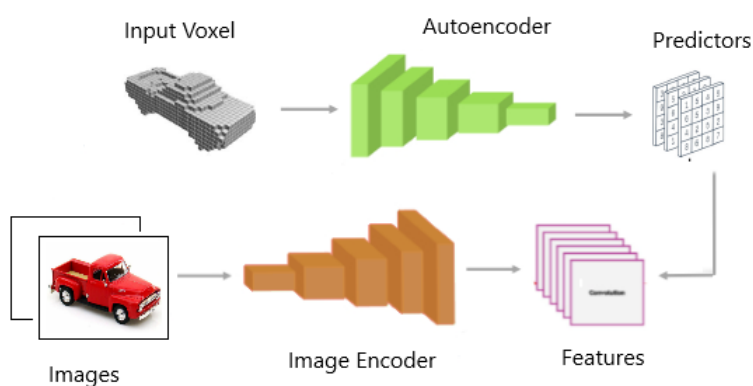


Figura 1.5: Processo di training della rete BSP-Net.

2. Nella seconda fase, training dell'immagine encoder, usando predittori ottenuti precedentemente e un dataset di immagini, tramite delle convoluzioni 2D, si ottengono delle feature che verranno confrontate con i predittori precedenti, con l'obiettivo di ottenere un match tra l'immagine bidimensionale e il modello 3D.

Considerando che la rete neurale BSP-Net non possiede il layer pooling, le immagini utilizzate per la fase di training saranno immagini con risoluzione 128x128px in scala di grigio e con un solo canale (L).

Concluso il processo di training, si passa alla fase di ricostruzione, in cui la rete neurale allenata prenderà in input un'immagine di un oggetto e fornirà, attraverso un processo inverso a quello del training, in output un modello 3D (vedi

figura 1.6). Bisogna fare in modo che le immagini in input abbiano la stessa risoluzione e lo stesso numero di canali delle immagini utilizzate in fase di training.

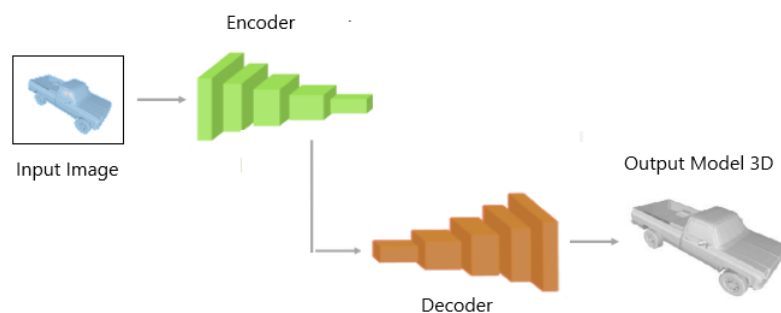


Figura 1.6: Processo di ricostruzione 3D a partire da un'immagine.

In questa ricerca, nella fase di ricostruzione dei modelli 3D, è stato utilizzato il dataset di immagini e modelli fornito dagli autori di *Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling* [34]. Questo particolare dataset fornisce sia le immagini reali dell'oggetto sia il corrispondente modello 3D, in quanto la maggior parte del dataset fornisce delle limitazioni. Alcuni di questi, da un lato offrono una gran varietà di modelli 3D, ma dall'altro questi associano ai modelli 3D delle immagini sintetiche e non reali; altri invece sono di piccole dimensioni. Pix3D offre circa 400 oggetti divisi in 9 categorie quali sedie, tavoli, letti, scrivanie, librerie, guardaroba, divani, strumenti e oggetti vari. Tuttavia, oltre ad offrire dei modelli 3D, per ogni modello Pix3D è possibile avere una serie di immagini del corrispondente oggetto, per un totale di circa 10.000 immagini. Grazie a questo tipo di dataset, è stato possibile valutare, utilizzando come input per la rete le immagini dell'oggetto e il corrispondente modello 3D originale (ground truth), la fedeltà o meno dell'oggetto ricostruito rispetto al suo corrispondente ground truth.

Per la realizzazione di tutti i vari dataset utilizzati durante il training, si è utilizzato il software Blender [3] e la pipeline di BlenderProc [10].

Blender è un software open source mirato alla modellazione 3D, animazione,

rigging, texturing, lighting, rendering, compositing e montaggio video. Dispone di molte funzionalità per simulazioni fisiche, particellari, fluidi e per mappature UV.

BlenderProc è una pipeline procedurale modulare open source che permette di generare immagini sintetiche per l'addestramento di reti neurali, offrendo una serie di moduli e funzioni sviluppati in linguaggio di programmazione Python [25]. Attraverso le funzionalità offerte da quest'ultimo, è possibile gestire: il caricamento di modelli 3D, l'applicazione di texture, luci e camere, ed inoltre, generare in output immagini a colori, depth map, normal map e maschere per la segmentazione delle stesse. I principali moduli sono mostrati in figura 1.7.



Figura 1.7: Principali moduli della pipeline di BlenderProc.

- Loader: caricamento di mesh, materiali, texture e oggetti all'interno della scena
- Camera: gestione del posizionamento della camera in un determinato punto, della profondità di campo, risoluzione e vari parametri relativi alla camera
- Lighting: gestione dell'illuminazione della scena
- Material: creazione o importazione di texture e materiali, con relativa gestione e applicazione ai vari oggetti
- Types: manipolazione dei vari elementi della scena
- Render: gestione delle impostazioni di rendering

L'intera parte relativa alla programmazione e i vari script sono stati sviluppati tramite il linguaggio di programmazione Python. Python è un linguaggio di programmazione dinamico, con licenza open source, utilizzato in diverse tipologie

di applicazioni, dal networking, al web, fino al machine learning.

Per la ricerca condotta nella tesi [22], la creazione dei dataset di immagini è avvenuta tramite tutte le librerie Python messe a disposizione da Blender e BlenderProc. Invece, per la gestione della rete neurale BPS-Net sono state utilizzate delle specifiche Python:

- NumPy: Numpy [18] è la principale libreria di calcolo scientifico, tramite cui è possibile effettuare qualsiasi tipo di operazione matematica complessa. Fornisce strumenti per aiutare a costruire array multidimensionali ed eseguire calcoli su dati memorizzati in essi, risolvere formule algebriche, eseguire operazioni statistiche e molto altro. Inoltre, supporta strumenti per l'integrazione di codice C/C++ e Fortran.
- PyTorch: PyTorch [26] è un framework per l'apprendimento automatico basato sulla libreria di machine learning Torch, la quale mette a disposizione diversi algoritmi avanzati per il deep learning. È utilizzato per elaborare e accelerare il calcolo scientifico dei tensori, ovvero degli array multidimensionali utilizzati nelle schede grafiche, riducendo i tempi di elaborazione. Grazie a ciò, Pytorch viene utilizzato anche come sostitutivo di NumPy riuscendo a sfruttare al meglio la potenza di calcolo della GPU (Graphics Processing Unit). Offre diversi moduli e librerie utili allo sviluppo di reti neurali profonde.
- Open3D: Open3D [19] è una libreria open source che supporta il rapido sviluppo di software che gestiscono dati 3D. Espone un insieme di strutture dati e algoritmi selezionati sia in C++ che in Python. Tuttavia, presenta diverse funzionalità come la creazione e l'elaborazione di strutture dati 3D, ricostruzione e visualizzazione di scene 3D, rendering di modelli 3D e supporto per l'apprendimento automatico 3D con PyTorch e TensorFlow.

1.2.2 Descrizione del lavoro

La pipeline di lavoro, adottata tramite software di modellazione 3D, per la generazione di dataset è suddivisa in diverse fasi.

1. Scelta della categoria di oggetti: scelta dei modelli utilizzati come base per la generazione del proprio dataset.

2. Creazione e/o raccolta di modelli 3D: fase di attenta ricerca o creazione di vari modelli 3D.
3. Creazione e/o raccolta di materiali: raccolta di materiali da applicare ai modelli, i quali vengono creati manualmente o presi da vari siti web.
4. Sistemazione dei modelli e applicazione dei materiali: normalizzazione dei modelli rispetto ad una specifica dimensione per rendere più semplice la gestione delle luci e camere, e successiva applicazione dei materiali con generazione di mappe UV.
5. Setup delle luci: creazione dell'illuminazione all'interno della scena, tramite la scelta adeguata di luci, colori e potenza luminosa, cercando di evitare ombre nette o riflessioni.
6. Setup della camera: impostazione dei parametri della camera, in base al tipo di utilizzo che si dovrà fare delle immagini ottenute dal processo di rendering, ed impostazione dei keyframe relativi alla posizione della camera.
7. Rendering: selezione della tipologia di rendering da utilizzare, relativa al modello preso in considerazione.

Dopo aver definito la pipeline di lavoro, si è cercato di capire quali feature potessero incidere sulla fase di training. In particolare, per tale ricerca, ci si è concentrati sui seguenti parametri:

- Dimensione dell'immagine
Analizzando il funzionamento delle diverse reti neurali, si è evidenziata una certa correlazione tra la dimensione dell'immagine e le prestazioni della rete. A tal proposito, si è deciso di puntare sulla creazione di dataset di immagini a differenti risoluzioni su cui effettuare i vari test.
- Texture e Materiali
Avendo la possibilità di poter gestire in diversi modi i materiali applicabili ad un oggetto, ci si è concentrati su quali di questi potessero migliorare o peggiorare la rete in fase di ricostruzione. Inoltre, nel caso di applicazione di determinate "mappe", si è arrivati alla conclusione che l'applicazione o meno della normal map ad un oggetto non influisce più di tanto sul processo di training e su una buona resa della ricostruzione del modello.
- Illuminazione
In fase di rendering, si è provato a capire quale fosse l'illuminazione più

adeguata del modello 3D e se la presenza di ombre e riflessioni potessero incidere in fase di training e durante il processo di ricostruzione. In particolare, sono state analizzate due differenti tipologie di illuminazione: illuminazione tramite una luce abbastanza omogenea e illuminazione tramite il modello *three point-lighting*⁴.

Definito tutto ciò, il primo compito fu quello di creare dei dataset di immagini sintetiche contenenti, oltre alle immagini dei vari modelli 3D, una rappresentazione in “*voxel*” di quest’ultimi. I modelli voxel, utilizzati in fase di training, sono stati ottenuti tramite un tool esterno chiamato *Binvox* [2] che permette di trasformare un modello 3D nel corrispondente modello in formato voxel. Successivamente, si è applicata l’intera pipeline di lavoro al dataset di immagini sintetiche creato.

In primo luogo, nella tesi citata precedentemente l’autore ha scelto di utilizzare circa 120 modelli, la maggior parte recuperati da BlenderKit [4] e dal sito web di Turbosquid [37]. Si è preferito utilizzare modelli in formato *.blend* già con texture e materiali applicati. Successivamente, si è passati ad una fase di verifica e correzione dei modelli, controllando l’orientamento delle normali e dell’oggetto rispetto agli assi di Blender e l’eventuale presenza di modificatori. Tali modelli poi sono stati normalizzati e scalati per essere contenuti all’interno di un “*empty*”, definito da un cubo unitario. Per l’assegnazione dei materiali, si è scelto di assegnare ad ogni oggetto dei materiali “vuoti”, definiti dal nome di una determinata categoria di materiali, per esempio “*Wood*” o “*Plastic*”. Inoltre, si è potuto far variare tali materiali per ogni diversa inquadratura tramite degli script, ottenendo così un dataset finale contenente circa 134 materiali; mentre, la gestione delle luci, camera e render è stata eseguita sfruttando le funzioni di BlenderProc. Per l’illuminazione sono stati utilizzati sia il modello di illuminazione omogenea che il modello *three-point lighting*; per il movimento della camera si è sviluppato uno script che fosse in grado di muovere la camera attorno all’oggetto, mantenendolo all’interno dell’inquadratura; per la fase di rendering, si son impostate

⁴Il modello *three point-lighting* è un setup di illuminazione comunemente utilizzato in fotografia, cinematografia e produzione video per ottenere una luce ben bilanciata e modellata su un soggetto. Questa tecnica coinvolge l’uso di tre fonti luminose principali posizionate per creare luce e ombre. Le principali fonti luminose sono: *key light* (luce principale), *fill light* (luce di riempimento) e *back light* (luce di contorno).

le dimensioni delle immagini in output (128x128px, 512x512px e 1024x1024px), il formato dell'immagine *.png* in output e la trasparenza del background.

In secondo luogo, si è passati alla gestione della rete neurale BPS-Net. I parametri principali che permettono il buon funzionamento di tale rete sono:

- **Pesi dell'autoencoder e dell'immagine encoder.** Generati nella fase di training e salvati poi all'interno di specifici file creati dalla stesa, sono necessari durante il processo di ricostruzione in quanto contengono tutte le informazioni utili che la rete ha appreso durante l'addestramento. Questi pesi vengono salvati ogni 10 o 100 epoche e fungono da "backup" dei progressi della rete. La scelta di salvare i pesi ogni 10 o 100 epoche è una pratica comune ma non standard. Salvare i pesi troppo frequentemente può aumentare il carico computazionale e l'uso dello spazio di archiviazione, d'altra parte se i pesi vengono salvati troppo raramente, vi è il rischio di perdere molte informazioni importanti in caso di crash del sistema.
- **Device da utilizzare.** BSP-Net, in questa ricerca, è stata utilizzata sfruttando la GPU.
- **Learning rate.** Il tasso di apprendimento utilizzato è stato quello proposto dagli autori di questa rete, ossia 0,0001.
- **Batch size.** Questo valore, ovvero la dimensione dei sottogruppi (batch) ottenuti dalla divisione dell'intero dataset utilizzato, dipende da diversi fattori come la dimensione del dataset, la disponibilità di memoria e la tipologia di rete. In questo caso, si è utilizzato un batch size pari a 16 a causa delle limitazioni di memoria della GPU.

Come già detto, BPS-Net sfrutta una doppia fase di training, una per l'autoencoder e una per l'immagine encoder. Avendo gli autori già fornito i pesi relativi ad un modello di autoencoder pre-allenato, si è evitata la fase di training dell'autoencoder, così da poter passare direttamente alla fase di estrazione dei predittori. Attraverso tale estrazione si è riusciti ad ottenere, per ogni modello voxel in input, un array delle feature fondamentali per il processo di ricostruzione.

Poiché la rete BSP-Net è stata progettata per lavorare con immagini di dimensione 128x128px in scala di grigi ad un solo canale, è stata trovata una soluzione per la gestione di immagini a differenti dimensioni. Essendo a conoscenza del fatto che la rete BSP-Net si basa sul modello di ResNet-18, ma senza livelli di pooling, è stato sufficiente aggiungere i due livelli di pooling, presenti invece in ResNet-18. Mentre, per il limite del numero di canali è bastato agire sul primo

livello convoluzionale cambiando il numero dei canali in input da 1 a 3.

Finito l'allenamento della rete, in cui i pesi e lo stato venivano salvati automaticamente all'interno di un file, si è passati alla fase di ricostruzione dei modelli 3D, appartenenti alle classi scelte, verificando, tramite il calcolo della *Chamfer Distance*[41], che i modelli ottenuti fossero più fedeli possibili all'originale. Per facilitare il calcolo della Chamfer Distance tra i modelli ricostruiti e i modelli ground truth, gli oggetti sono stati ricostruiti come nuvole di punti.

Il processo di estrazione dei predittori, il training dell'immagine encoder, la ricostruzione e il calcolo della metrica è stato ripetuto per ogni singolo dataset di immagini sintetiche.

La risoluzione base delle immagini usate nel training per l'analisi è stata 128x128px; in particolare, per la rete BSP-Net, si è utilizzata anche la risoluzione 224x224px. Per quanto riguarda la distanza camera-oggetto, si è pensato di generare dei dataset di immagini in cui la camera, oltre a muoversi attorno all'oggetto, cambiasse la sua distanza da esso in modo casuale. Per i materiali si sono fatti dei render sia per oggetti con materiale applicato che senza; dato che i vari materiali sono costituiti da diverse mappe, si sono anche generati dei dataset in cui ogni volta veniva rimossa una singola mappa fino, ad avere solamente una texture col colore del materiale. Per le luci, si sono definiti due modelli di illuminazione: un modello di illuminazione omogenea, andando a ridurre il più possibile le zone d'ombra, e un modello three-point lighting, con due luci poste sul modello 3D ed una dietro. In entrambi questi modelli la luce ha un'intensità fissa, quindi si è creato un dataset dove per ogni singola immagine l'intensità della luce era differente.

Inoltre, si è dovuto identificare quale fosse il metodo più efficace per eseguire una valutazione in termini di precisione nella ricostruzione di un modello 3D. Per valutare l'effettiva fedeltà del modello 3D ricostruito dalle rete neurale rispetto all'originale, si è scelto di utilizzare la Chamfer Distance. Più basso è il valore della distanza, minori saranno le differenze tra le due nuvole di punti, quindi tra i due modelli 3D. La rete neurale BSP-Net fornisce in output oltre che le mesh 3D anche le nuvole di punti degli oggetti ricostruiti.

1.2.3 Risultati ottenuti

Nello specifico, i test sono stati effettuati sulla base di tre dataset di immagini:

- Il primo contenente delle immagini sintetiche mantenendo fissa la distanza della camera dal modello 3D.
- Il secondo contenente delle immagini ottenute applicando ai vari modelli 3D gli stessi materiali del primo dataset, ma facendo variare la distanza della camera.
- Il terzo contenente delle immagini utilizzando le pose della camera del secondo dataset, ma rimuovendo dai modelli 3D tutti i materiali.

Considerando questi tre dataset, si è analizzato come variano le prestazioni di ricostruzione in funzione di alcuni parametri fondamentali che influiscono sul training.

a) **Risoluzione dell'immagine**

I primi esperimenti sono stati fatti in riferimento alla dimensione delle immagini. Dai risultati ottenuti, si è osservato che all'aumentare della risoluzione diminuisce il valore della Chamfer Distance e, di conseguenza, la ricostruzione dell'oggetto risulta essere migliore. In particolare, quando si aumenta la risoluzione da 128x128px a 224x224px, si osserva un miglioramento della Chamfer Distance, la quale diventa più eterogenea aumentando ulteriormente la risoluzione da 224x224px a 512x512px. Al contrario, quando si passa da una risoluzione di 512x512px a 1024x1024px, i risultati della Chamfer Distance mostrano una maggiore variabilità tra i diversi test effettuati, e in alcuni casi la qualità della ricostruzione potrebbe addirittura lievemente peggiorare.

Tuttavia, al crescere della dimensione delle immagini in input aumenta la matrice delle feature. Nei primi tre esperimenti, il numero di epoche necessarie per concludere una sessione di training è stato in media di 260 per immagini di dimensione 128x128px, 350 per immagini di dimensione 224x224px, 545 per immagini di dimensione 512x512px e di 709 per immagini di dimensioni 1024x1024px.

b) **Distanza della camera**

Altri risultati si sono ottenuti andando ad agire sulla posizione della camera. Facendo variare la distanza della camera rispetto all'oggetto, la ricostruzione migliorava rispetto all'utilizzo di una camera fissa posta ad una determinata distanza. Questi risultati sono dovuti a diversi motivi quali l'utilizzo

di immagini reali in fase di ricostruzione e la forma semplice o complessa dell'oggetto.

c) **Materiali e texture**

Nel caso dei materiali, la presenza di quest'ultimo migliora i risultati rispetto ad utilizzare immagini di oggetti senza materiale.

d) **Illuminazione**

Invece, nel caso dell'illuminazione, un'illuminazione omogenea adeguata migliora le prestazioni della rete in fase di ricostruzione rispetto magari ad un'illuminazione più realistica come three-point lighting. Tuttavia, aumentando la risoluzione dell'immagine e utilizzando una diversa illuminazione, la ricostruzione peggiorava a causa di ombre o riflessioni.

e) **Scala di grigi**

Per quanto riguarda dataset di immagini in scala di grigi, i risultati ottenuti sono stati peggiori rispetto all'utilizzo di immagini a colore RGB, anche se i tempi di training per epoca risultano più veloci. Ciò è dovuto al fatto che nel caso di immagini ad un solo canale, le convoluzioni non vengono effettuate su tre canali (RGB), ma solamente su uno (L).

La tabella 1.1 mostra un riepilogo dei risultati ottenuti in base al variare dei parametri.

Parametri	Chamfer Distance	Qualità ricostruzione
512x512	Eterogenea +3.3 - 4.4%	MIGLIORE
1024x1024	Varia	PEGGIORE
Distanza varia	Migliora +50%	MIGLIORE
Presenza materiale	Migliora +10%	MIGLIORE
Assenza materiale	Elevata	PEGGIORE
Illuminazione omogenea	Migliora +7%	MIGLIORE
Illuminazione realistica	Alta	PEGGIORE
Scala di grigi	Elevata	PEGGIORE

Tabella 1.1: Variazione delle prestazioni di ricostruzione in funzione dei vari parametri.

Inoltre, altri elementi che potrebbero influire sul training della rete sono l'impiego della CPU (Central Processing Unit) o GPU, le librerie utilizzate e le rispettive versioni.

1.3 Obiettivo della tesi

L'obiettivo della presente tesi consiste nell'eseguire un'analisi comparativa accurata della precisione di ricostruzione attraverso l'utilizzo della metrica nota come Chamfer Distance su modelli tridimensionali. Questi modelli sono ottenuti a partire da un insieme di dati di immagini bidimensionali precedentemente impiegati nello studio condotto dall'ing. Piparo.

Nello specifico, l'obiettivo di tale valutazione è stabilire l'adeguatezza dell'approccio metodologico delineato dall'ingegnere per la rete neurale BSP-Net, estendendo tale metodologia al di là del contesto specifico dell'architettura base di BPS-Net, e valutando la possibilità di estenderla ad altre reti neurali con architetture diverse.

Nella fase iniziale del lavoro si è condotta un'analisi dettagliata di varie architetture di reti neurali, diverse dalla struttura BSP-Net ma tutte indirizzate alla ricostruzione tridimensionale partendo da rappresentazioni bidimensionali. Nello specifico, lo studio delle reti neurali richiedeva l'osservazione di requisiti precisi:

- Addestramento della rete utilizzando singole immagini bidimensionali RGB al fine di realizzare una ricostruzione da una sola vista - *Single-view Reconstruction*.
- Operatività all'interno del contesto delle categorie di oggetti presenti all'interno di ShapeNet.
- Assenza di legami con l'architettura di ResNet-18 o sue varianti.
- Capacità di esecuzione non limitata unicamente a immagini con sfondi chiari.

Una volta completata l'analisi comparativa delle diverse reti, il focus si è spostato sulla selezione di un modello specifico per il caso di studio. Nello specifico, è stata scelta la rete neurale *Pix2Vox*. *Pix2Vox* si caratterizza per la sua abilità di operare su immagini bidimensionali con una dimensione di 224x224 pixel, la quale è stata addestrata sulle categorie di oggetti presenti all'interno del dataset ShapeNet.

In una fase successiva, si è passati alla configurazione della rete scelta focalizzandosi, in particolare, sull'addestramento attraverso l'impiego del dataset di immagini precedentemente usato nella tesi di riferimento. Questo dataset è stato generato mediante la raccolta di modelli tridimensionali rappresentativi di varie categorie di oggetti, scelti con l'intenzione di impiegarli nell'addestramento della rete neurale.

Nel corso di diversi esperimenti, è emerso che numerosi fattori sono in grado di influenzare il procedimento di addestramento. Tra questi fattori si ha la risoluzione dell'immagine, le condizioni di illuminazione, la presenza o meno di texture e altri elementi correlati.

Basandosi su queste informazioni, si è proceduto a suddividere l'insieme di dati in categorie corrispondenti a specifiche tipologie di oggetti, ognuna caratterizzata da determinate proprietà.

Prima di condurre le prove definitive, la rete è stata allenata inizialmente utilizzando il dataset di riferimento ma ridimensionando le immagini, in esso contenute, a una risoluzione di 224x224px, corrispondente alla dimensione operativa della rete. Solo successivamente, la rete è stata sottoposta a ulteriori cicli di addestramento riducendo le dimensioni del dataset a 128x128px.

Infine, oltre a monitorare i tempi richiesti dalla rete neurale per completare le

fasi di addestramento in relazione alle diverse iterazioni condotte, è stata fatta un'analisi comparativa dei risultati ottenuti dalle diverse prove. Tuttavia, l'obiettivo principale di questo confronto non è stato quello di analizzare i valori assoluti della metrica di valutazione, bensì di controllare l'andamento di quest'ultima al variare dei differenti parametri quali la risoluzione dell'immagine, la distanza dalla telecamera, l'illuminazione della scena e la presenza o assenza di materiale.

2 Stato dell'arte

2.1 Reti neurali per la ricostruzione di elementi 3D

La ricostruzione di oggetti 3D è una parte fondamentale dell'elaborazione delle immagini e della visione artificiale. Questo processo, che consiste nella creazione di modelli 3D dettagliati di oggetti a partire da dati bidimensionali, è utilizzato all'interno di una vasta gamma di applicazioni come la robotica, la progettazione assistita da computer (CAD⁵), la realtà virtuale (VR), la realtà aumentata (AR), la medicina, la grafica computerizzata e molte altre. L'obiettivo finale della ricostruzione 3D è quello di creare dei modelli che rappresentino fedelmente la forma e la struttura degli oggetti.

Oltre ad avere delle tecniche impiegate nella realizzazione di una copia virtuale di un determinato oggetto reale, come la fotogrammetria, che sfrutta o immagini digitali per estrarre informazioni tridimensionali, e la scansione LiDAR che utilizza impulsi laser, esistono altri metodi tradizionali per la ricostruzione 3D basati su tecniche matematiche e algoritmi. I più comuni, mostrati in figura 2.1, sono:

- Structure from Motion (SfM) [21]: tecnica che stima la posizione della fotocamera e la struttura 3D di un oggetto utilizzando una serie di immagini catturate da diverse angolazioni. Si basa sulla rilevazione delle corrispondenze tra punti chiave (feature) nelle diverse immagini per stimare le posizioni 3D dei punti.
- Simultaneous Localization and Mapping (SLAM) [12]: utilizzato principalmente nella robotica, si basa sulla combinazione di dati di sensori per aggiornare la posizione stimata di un robot e costruire la mappa dell'ambiente circostante.

⁵CAD è un acronimo di *computer-aided design* volto all'utilizzo di software computerizzati per supportare e semplificare il processo di progettazione e disegno in vari campi professionali come l'architettura, l'ingegneria macchine e l'industria.

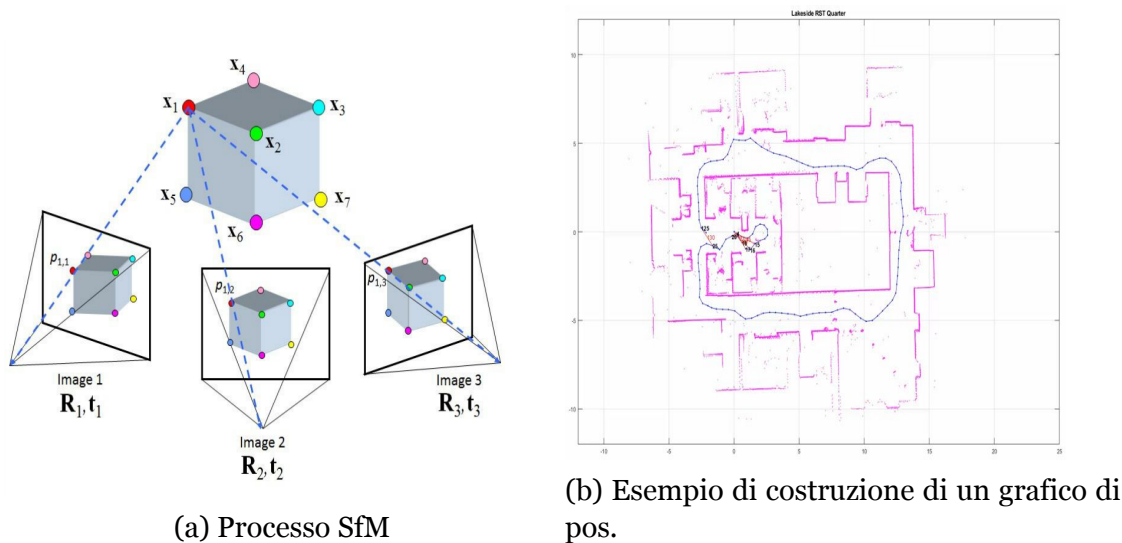


Figura 2.1: Rappresentazione di due tecniche comuni.

Entrambi questi metodi però presentano delle limitazioni quando si tratta di stabilire corrispondenze tra feature nel caso di viste multiple separate da un ampio margine a causa di locali cambiamenti nell'aspetto degli oggetti o occlusioni. Di conseguenza, l'uso di reti neurali e approcci basati sull'apprendimento automatico diventano l'alternativa per affrontare tali problemi.

Le reti neurali per la ricostruzione 3D possono variare in termini di architettura, tipo di dati in input, tecniche di addestramento, nella rappresentazione dell'output e altro ancora. Alcune reti, come *3D-GAN* [6], sono progettate per produrre direttamente una rappresentazione tridimensionale dettagliata di un oggetto sotto forma di una mesh 3D, la quale consiste in un insieme di poligoni rappresentanti la superficie dell'oggetto. Altre (*PointNet* [27]) generano una nuvola di punti o *point cloud*, ovvero un insieme di coordinate tridimensionali rappresentanti i punti che compongono la superficie dell'oggetto. Altre, come *Occupancy Networks* [16] e *3D-R2N2* [9], producono una rappresentazione tridimensionale sotto forma di una griglia di *voxel*, dove ogni voxel è una porzione 3D dello spazio che può essere occupato (valore 1) o vuoto (valore 0). Le diverse modalità di rappresentazione sono mostrate in figura 2.2.

In particolare, la rete Pix2vox utilizza un approccio *Encoder-Decoder* che genera come output una rappresentazione voxel dell'oggetto. Nella fase di encoding, l'immagine 2D viene elaborata e trasformata, tramite una serie di strati convo-

luzionali, in un vettore di caratteristiche. Nella fase di decoding, tramite strati di convoluzione tridimensionali trasposti, la rappresentazione di feature viene utilizzata per generare la rappresentazione 3D dell'oggetto sotto forma di voxel ad alta risoluzione. Inoltre, per migliorare la qualità di ricostruzione, l'architettura integra altri due moduli: Context-aware Fusion Module e Refiner.

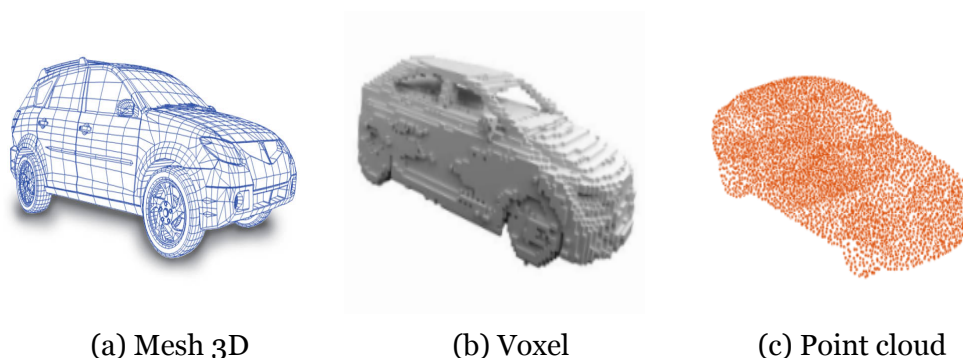


Figura 2.2: Diverse rappresentazioni di un modello 3D.

2.2 Principali dataset di immagini

La maggior parte delle reti neurali dedicate alla ricostruzione di modelli 3D, durante la fase di training, utilizza dataset di immagini contenenti diverse classi di oggetti. Ciò risulta fondamentale affinché ogni rete neurale possa essere in grado di apprendere rappresentazioni tridimensionali di oggetti provenienti da varie categorie. L'utilizzo di dataset diversificati è fondamentale e aiuta la rete a sviluppare una comprensione maggiore delle caratteristiche e delle forme degli oggetti. Addestrare la rete su diverse classi di oggetti aventi strutture e dettagli distinti, la rende più robusta nel riconoscimento e ne migliora la qualità nella ricostruzione.

A seconda degli obiettivi specifici e delle caratteristiche del modello, le tipologie di immagini utilizzate per il processo di training sono differenti. Alcune reti neurali vengono allenate su dataset generici contenenti una vasta gamma di oggetti in vari contesti come immagini di strada, interni, volti e altro ancora. Altre reti, invece, sono progettate per una classe specifica di oggetti per far sì che queste ultime vengano ottimizzate per riconoscere e ricostruire specifiche caratteristiche

e dettagli della classe presa in considerazione. Per adattarsi alle diverse esigenze delle reti neurali, si utilizzano immagini R (rosso), G (verde), B (blu) per catturare le informazioni di colore e luminosità e/o immagini RGB-D (depth), dove D rappresenta la componente di profondità. A volte, le immagini vengono trasformate o processate prima del loro utilizzo, ad esempio le immagini RGB possono essere convertite in scala di grigi (figura 2.3).

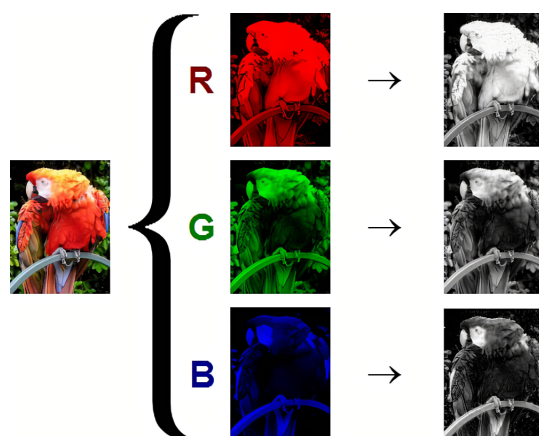


Figura 2.3: Partendo da un'immagine a colori naturali, la colonna a sinistra mostra i canali di colore isolati RGB, mentre a destra ci sono le loro equivalenze in scala di grigi.

Un altro elemento chiave delle immagini che ha un impatto significativo sul training è la risoluzione, la quale risulta variabile da rete a rete. Immagini ad alta risoluzione possono dare come risultato dei modelli più complessi, poiché le reti devono analizzare più dettagli richiedendo così l'utilizzo di più risorse computazionali e potenza di calcolo. Al contrario, con immagini a bassa risoluzione si potrebbe avere una perdita di dettagli.

Di conseguenza, un approccio comune è quello di iniziare con basse risoluzioni, per i modelli iniziali, e aumentare gradualmente la risoluzione man mano che la complessità e le risorse lo consentono. In generale, molti dataset utilizzati per la ricostruzione 3D contengono principalmente immagini sintetiche invece che "reali". Questi dati sintetici sono creati utilizzando software di rendering e modelli 3D per simulare l'aspetto degli oggetti da diverse angolazioni e condizioni di illuminazione. Tale approccio presenta diversi vantaggi:

- Maggior controllo e accuratezza sugli oggetti rappresentati

- Forniscono dati annotati in modo semplice e richiedono minor tempo rispetto a grandi quantità di dati reali
- Generare un numero maggiore di campione di addestramento con variazioni controllate come cambiamenti di illuminazione, angoli di visuale e sfondi
- Ottenere una gran quantità di immagini a partire da un unico modello 3D di un oggetto reale

Tuttavia, l'uso di dataset sintetici potrebbe presentare delle immagini non perfettamente realistiche, poiché non catturano al meglio alcune sfumature e variazioni presenti nelle immagini reali.

Ci sono diversi metodi e strumenti per generare dataset di immagini sintetiche. Tra questi si hanno:

- Software di rendering 3D come Blender [3], Maya [15], Unity [38], che permettono di creare modelli 3D dettagliati e renderizzarli da diverse angolazioni
- Approccio procedurale che coinvolge l'utilizzo di algoritmi per generare oggetti e ambienti
- Utilizzo di modelli 3D preesistenti disponibili gratuitamente o a pagamento su vari siti Internet
- Dataset sintetici già esistenti disponibili gratuitamente
- Software che offrono plugin e capacità di scripting per automatizzare la generazione di immagini sintetiche, consentendo di generare grandi quantità di dati in modo efficiente

Creare modelli 3D da zero può richiedere competenze avanzate in modellazione 3D e un lungo lavoro di perfezione e accuratezza. Per tale motivo l'utilizzo di modelli 3D già esistenti e completi di texture e materiali permette di accedere ad una vasta gamma di oggetti e scene, riducendo tempo e lavoro nella modellazione e nella creazione di materiali. Esistono diversi siti online da dove poter scaricare diversi modelli 3D pronti all'uso (figura 2.4), tra i principali:

CGTrader [5]: mercato per l'acquisto e la vendita di progetti 3D e file CAD.

Turbosquid [37]: grande mercato online contenente migliaia di modelli 3D per diverse categorie. Per ogni categoria, inoltre, sono presenti dei filtri per poter

selezionare il prezzo, il tipo di formato, la qualità e molto altro.

Sketchfab [33]: piattaforma di condivisione di modelli 3D interattivi, dove poter sfogliare, visualizzare e scaricare modelli in diversi formati.

BlenderKit [4]: add-on integrato nel software di modellazione Blender dove gli utenti possono cercare, visualizzare e scaricare una vasta gamma di risorse come modelli 3D, materiali, oggetti, texture e altro ancora.

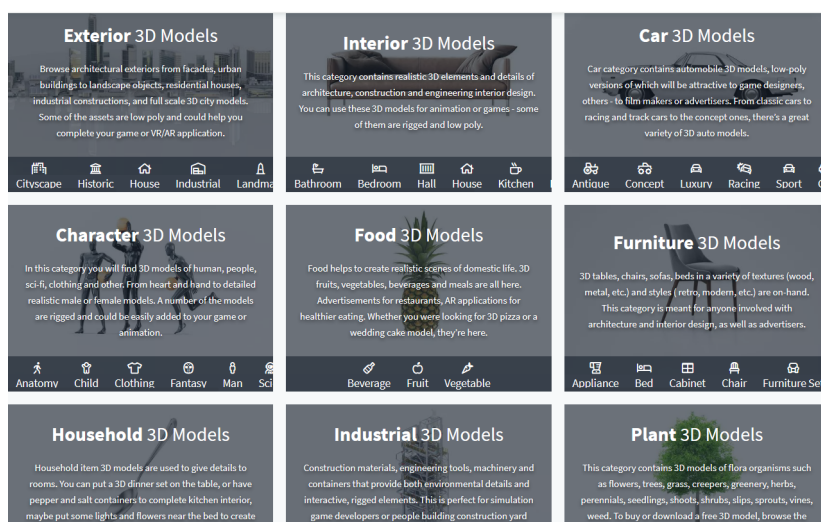


Figura 2.4: Alcune tipologie di modelli 3D pronti all'uso.

Tuttavia, il processo per la creazione di dataset di immagini sintetiche non segue una regola fissa. Molti ricercatori creano dataset sintetici su misura per dimostrare l'efficacia dei loro algoritmi o modelli e avere un controllo maggiore sui dati. Mentre, i dataset pubblici generalmente vengono utilizzati dai ricercatori come benchmark per valutare e confrontare diversi algoritmi e modelli, risparmiando tempo e sforzi. Ci sono diversi dataset pubblici disponibili che coprono una vasta gamma di ambiti e applicazioni nell'ambito dell'elaborazione delle immagini:

Shapenet [7]: grande archivio di più di 50.000 modelli 3D che copre 55 categorie, utilizzato principalmente per la ricerca nell'ambito della visione computerizzata e dell'apprendimento automatico e per attività come il riconoscimento

degli oggetti, la segmentazione semantica e la ricostruzione 3D.

Pix3D [35]: dataset di circa 400 modelli 3D di vari oggetti, utilizzato per sviluppare algoritmi che possono allineare oggetti in immagini 2D con i loro modelli 3D corrispondenti.

ImageNet [30]: uno dei più grandi dataset di immagini sintetiche, utilizzato per l'addestramento e la valutazione di reti neurali profonde per una varietà di compiti, inclusa la classificazione di immagini e la rilevazione di oggetti.

Pascal3D+ [42]: dataset contenente immagini reali con un allineamento approssimativo tra immagini 2D e forme 3D, in quanto gli oggetti nelle immagini sono abbinate a un set predefinito di modelli CAD e non al loro set reale. È utilizzato per una serie di applicazioni, tra cui il rilevamento di oggetti 3D, la stima della posa, la correzione delle occlusioni e la comprensione delle relazioni spaziali.

2.3 Altri studi nella generazione di dataset

L'utilizzo di dati sintetici per l'addestramento è diventato sempre più popolare. Grazie alle tecnologie di rendering ed alla possibilità di avere a disposizione diversi modelli 3D, è possibile generare velocemente e a basso costo una molteplicità di immagini sintetiche. Tuttavia, l'uso di dati sintetici presenta una limitazione: il divario nell'aspetto tra le immagini reali e quelle renderizzate causa una riduzione nelle prestazioni del modello addestrato su immagini sintetiche quando viene testato su immagini reali. Questo fenomeno è noto come *"domain gap"*. Il *"domain gap"* deriva dall'incapacità di replicare fedelmente tutti gli aspetti di una telecamera del mondo reale nei dati sintetici.

Per colmare questo divario, alcuni ricercatori hanno esplorato diverse strategie [40]. Alcuni hanno parlato di randomizzazione del dominio, una tecnica che prevede l'introduzione di casualità o variazione in vari aspetti (condizione di illuminazione, pose degli oggetti, sfondi e altro) del processo di generazione dei dati sintetici al fine di rendere i dati sintetici più simili ai dati reali. Altri hanno utilizzato il trasferimento di apprendimento, ovvero invece di addestrare il modello da zero si utilizzano pesi preaddestrati su dati reali per poi adattare il modello al contesto sintetico. Nel caso in cui i dati sintetici non riescano però a catturare la complessità dei dati reali, entra in gioco l'effetto dell'aggiunta di immagini reali durante l'addestramento su dati sintetici, per far sì che il modello si

adatti meglio alle caratteristiche specifiche dei dati reali.

L'obiettivo è quello di migliorare la capacità di apprendimento automatico dei modelli addestrati su dati sintetici, consentendo loro di essere più performanti e affidabili quando applicati a scenari reali.

3 Tecnologie utilizzate

3.1 Reti convoluzionali

Come già discusso in precedenza, le reti neurali convoluzionali, spesso indicate anche come ConvNets o CNNs, sono un tipo di architettura di rete neurale artificiale specializzata nella lavorazione di dati con una struttura di griglia, come immagini o dati bidimensionali (figura 3.1). Queste offrono un approccio scalabile e sfruttano principi dell'algebra lineare per identificare i modelli all'interno di un'immagine.

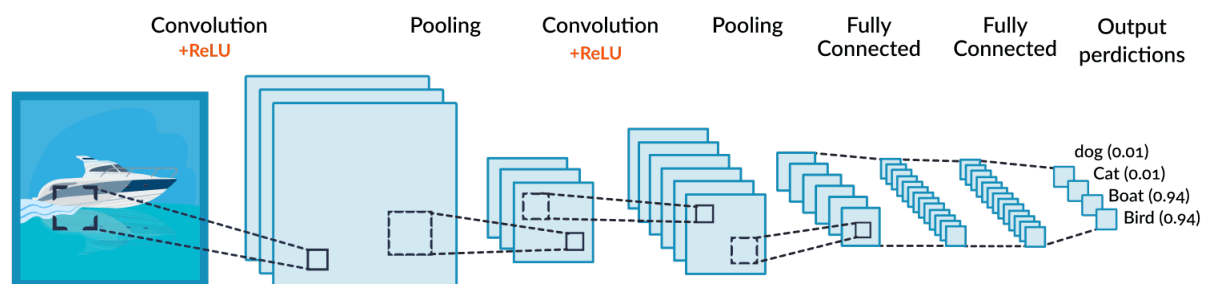


Figura 3.1: Architettura base di una rete CNN.

Prima di selezionare il caso studio per questa ricerca, si è condotto uno studio approfondito sull'architettura e gli aspetti correlati di diverse reti neurali convoluzionali. L'obiettivo principale è stato quello di allontanarsi dal contesto specifico della rete BSP-Net e da qualsiasi derivato del modello ResNet-18, per far sì che la metodologia del lavoro precedente [22] potesse essere applicata in modo più generale ed essere estesa ad altre reti neurali volte alla ricostruzione 3D.

Durante questa fase di analisi, ci si è concentrati su reti neurali che soddisfacessero determinate specifiche:

- **Ricostruzione 3D da singola immagine RGB:** Era essenziale che la rete fosse in grado di generare una ricostruzione 3D a partire da una singola immagine a colori RGB.
- **Nessun addestramento su altri dati:** Son state ricercate reti neurali che non richiedessero un addestramento su dataset esterni, ma potessero essere utilizzate con successo con il dataset disponibile.
- **Utilizzo di immagini sintetiche e dataset di Shapenet:** Era importante che la rete fosse in grado di operare con immagini sintetiche e fosse

compatibile con le classi di oggetti di Shapenet, il che era fondamentale per la nostra ricerca.

- **Non limitata a uno sfondo chiaro:** Le reti neurali considerate non dovevano essere vincolate all'utilizzo di sfondi chiari nelle immagini di input.
- **Indipendenza dalla struttura di ResNet e derivati:** Son state selezionate reti neurali che avessero una struttura diversa da ResNet e le sue varianti, al fine di esplorare nuove prospettive e approcci.
- **Disponibilità di codice e dataset di esempio scaricabile:** Si è preferito scegliere reti neurali per le quali fosse disponibile del codice sorgente e dataset di esempio facilmente accessibili per la nostra ricerca.

Dopo aver attentamente esaminato le reti neurali che soddisfacevano tali criteri e aver condotto un'analisi dettagliata sul loro funzionamento, si è selezionata *Pix2Vox* [43] come rete di riferimento per la nostra tesi.

Di seguito verranno esaminate alcune delle reti neurali prese in considerazione per questa ricerca.

3.1.1 CoreNet

“CoReNet: Coherent 3D scene reconstruction from a single RGB image” [24] è un modello di apprendimento profondo sviluppato per la ricostruzione 3D *Single-view* da un'immagine RGB. Questo modello si basa su un'architettura di codificatore-decodificatore (figura 3.2) e presenta tre estensioni chiave: connessioni di skip ray-traced per poter propagare all'output le informazioni locali del volume 3D; una rappresentazione ibrida del volume 3D per poter gestire sia le informazioni globali che i dettagli fini; utilizzo di una perdita di ricostruzione per catturare la geometria complessiva dell'oggetto.

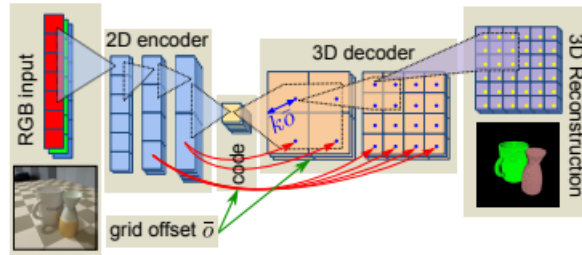


Figura 3.2: Architettura della rete CoReNet. Il dato viene propagato dall'encoder al decoder tramite le connessioni di skip-ray traced (freccie rosse).

Il modello è stato validato sperimentalmente sia su dati sintetici provenienti da ShapeNet che su immagini reali provenienti da Pix3D, con sfondi complessi e ombre.

Tuttavia, oltre a gestire oggetti singoli, il modello è stato adattato anche per la ricostruzione di più oggetti da un'unica immagine. Questi oggetti vengono ricostruiti congiuntamente in un unico passaggio, producendo una ricostruzione coerente in cui tutti gli oggetti esistono in un singolo sistema di coordinate tridimensionale rispetto alla telecamera, senza sovrapposizioni all'interno dello spazio 3D.

3.1.2 DISN

“DISN: Deep Implicit Surface Network for High-quality Single-view 3D Reconstruction” [44] è una rete neurale che ha la capacità di ricostruire una forma 3D catturando sia l'intera struttura dell'oggetto che i dettagli fini, a partire da un'immagine 2D.

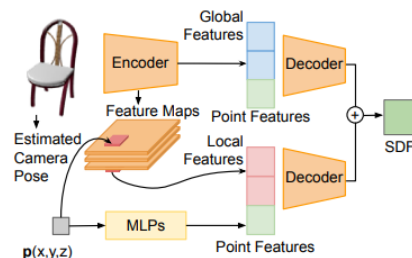


Figura 3.3: Funzionamento della rete DISN.

Poiché risulta difficile acquisire i dettagli sottili di una forma 3D, si è introdotto un metodo di estrazione di caratteristiche locali. Come mostrato in figura 3.3, partendo da un punto 3D proiettato su una posizione 2D nel piano dell'immagine, tramite i parametri stimati della camera, vengono estratte le caratteristiche dalle mappe di caratteristiche (feature map) corrispondenti alla posizione 2D proiettata, le quali vengono combinate per ottenere le caratteristiche locali dell'immagine. Successivamente, due decoder prendono in input le caratteristiche globali e locali dell'immagine con le caratteristiche dei punti e effettuano una previsione SDF⁶.

Prevedendo tale valore, l'estrazione delle caratteristiche locali aiuta a recuperare questi dettagli mancanti della forma dell'oggetto, in quanto attraverso le caratteristiche globali la rete è in grado di prevedere solo la forma complessiva.

La rete DISN effettua sia una ricostruzione *Single-view* che *Multi-view*, partendo da un addestramento sulle categorie di oggetti presenti nel dataset di ShapeNet Core. Tuttavia, non ricostruisce direttamente mesh 3D, ma il risultato è una superficie implicita, ovvero una rappresentazione matematica di una superficie senza la necessità di una mesh o topologia esplicita. Nonostante la qualità di ricostruzione risulta elevata, presenta il limite di poter trattare solamente oggetti con sfondo chiaro.

3.1.3 D²IM-Net

“*D²IM-Net: Learning Detail Disentangled Implicit Fields from Single Images*” [13] è una rete progettata per una ricostruzione 3D *Single-view* che possa recuperare la struttura topologica dell'immagine di input e i dettagli di superficie.

⁶*Signed Distance Field* (SDF) rappresenta la distanza da un punto 3D alla superficie di un oggetto, con valori positivi che indicano punti al di fuori dell'oggetto e valori negativi che indicano punti all'interno dell'oggetto.

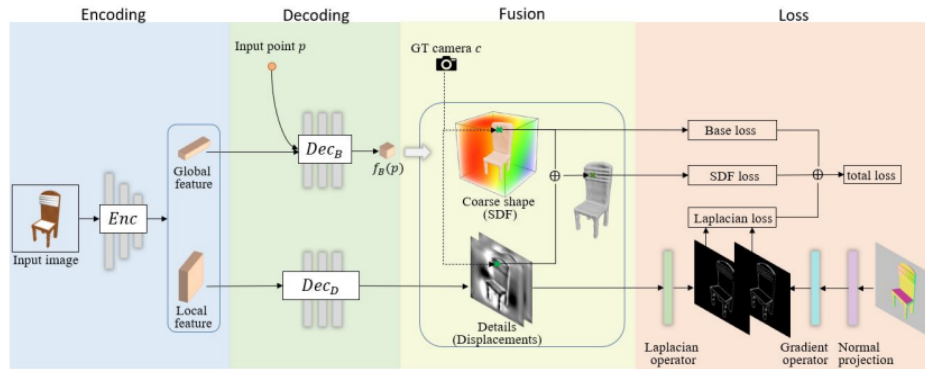


Figura 3.4: Struttura della rete D^2IM -Net composta da tre fasi.

Data una singola immagine RGB in input, la rete estrae da essa informazioni sia a livello globale che locale, le quali verranno utilizzate da due diversi decoder (figura 3.4). Il primo decoder, chiamato *base decoder*, utilizza le informazioni globali per ricostruire una rappresentazione approssimativa della forma 3D. Il secondo decoder, detto *detail decoder*, si concentra sulle informazioni locali e le utilizza per ricostruire due mappe di spostamento o *displacement map*, una per la parte anteriore e una per la parte posteriore dell'oggetto catturato.

La ricostruzione 3D finale è ottenuta combinando la forma di base con le mappe di spostamento. Inoltre, vi sono due funzioni di perdita, *base loss* e *SDF loss*, che garantiscono il recupero accurato della forma grossolana, della struttura complessiva e dei dettagli superficiali, e un termine *Laplaciano* per migliorare la qualità della ricostruzione.

D^2IM -Net viene addestrata sul dataset di Shapenet, contenente una vasta gamma di forme e oggetti provenienti da diverse categorie. Tuttavia, essendo l'obiettivo quello di recuperare dettagli geometrici su piccola scala, questo modello è in grado di catturare dettagli solo su superfici relativamente piatte, senza essere in grado di gestire dettagli su superfici molto curve.

3.1.4 Image2Mesh

“*Image2Mesh: A Learning Framework for Single Image 3D Reconstruction*” [23] è un framework basato su un grafico che incorpora mesh 3D in uno spazio basso-dimensionale e che consente ricostruzioni con un alto livello di dettagli.

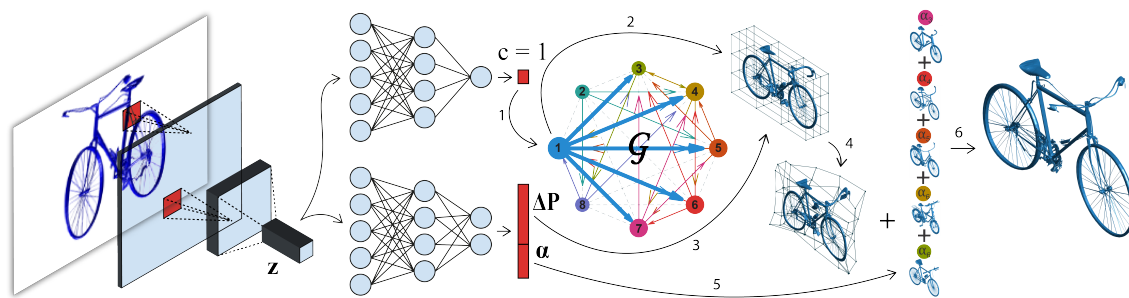


Figura 3.5: Architettura della rete neurale Image2Mesh.

È basato su un approccio grafico per la ricostruzione di mesh 3D a partire da singole immagini. Partendo da una singola immagine in input, si utilizza un *autoencoder convoluzionale* per estrarre le informazioni chiave dall'immagine. Selezionato il modello 3D appropriato da un grafo embedding G , lo si deforma in base ai parametri stimati FFD⁷ e, attraverso una combinazione lineare dei nodi, si combinano le informazioni del grafo per ottenere una ricostruzione 3D dettagliata dell'oggetto presente nell'immagine (figura 3.5). Inoltre, l'utilizzo del grafo è cruciale per stabilire le corrispondenze dense o meno tra il modello 3D stimato e l'oggetto nell'immagine.

Per addestrare Image2Mesh, gli autori hanno optato per la generazione di modelli 3D sintetici tramite l'utilizzo di grafi già esistenti, dove ogni grafo contiene una serie di modelli CAD campionati da ShapeNet. Inoltre, per ogni modello 3D sintetizzato, si sono create una serie di immagini 2D da diverse prospettive e sfondi bianchi.

Dalla qualità di ricostruzione e dai risultati ottenuti, si è dimostrato come tale rete neurale lavora con mesh poligonali dense anziché rappresentazioni volumetriche o point cloud.

⁷La *Free-Form Deformation* (FFD) è una tecnica utilizzata nella computer grafica e modellazione 3D per deformare e manipolare gli oggetti tridimensionali in modo flessibile, ottenendo così la forma desiderata.

3.1.5 Sym3DNet

“*Sym3DNet: Symmetric 3D Prior Network for Single-View 3D Reconstruction*” [32] è un modello di rete *Single-view Reconstruction* che sfrutta la struttura di simmetria a riflessione 3D di un oggetto e si concentra sulla previsione di forme 3D voxelizzate da un’immagine 2D.

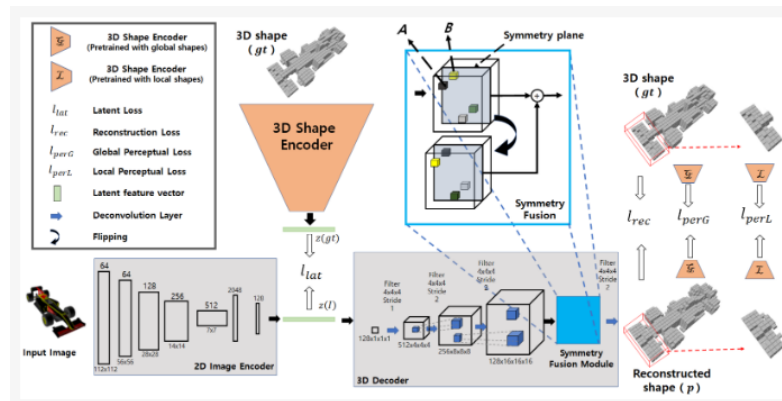


Figura 3.6: Funzionamento della rete Sym3DNet.

Come si vede in figura 3.6, Sym3DNet è composto da due rami di encoder addestrati congiuntamente, un encoder di forma 3D e un encoder di immagine 2D, e da un decoder 3D è collegato agli encoder in modo che il volume 3D di input e l’immagine 2D siano rappresentati nello stesso spazio latente e ricostruiscano il volume 3D. Inoltre, viene introdotto uno schema di condivisione delle caratteristiche di simmetria, *Symmetry Fusion Module*, e una perdita percettiva. Tramite il modulo di simmetria, le informazioni rilevanti vengono condivise tra le diverse parti di un oggetto, consentendo alla rete di ricostruire parti mancanti di un oggetto basandosi sulla simmetria presente in altre parti dell’oggetto. Mentre la perdita percettiva, in termini sia globali che locali, recupera la naturalezza della forma 3D e i dettagli.

Sym3DNet è stato valutato sperimentalmente su dati sintetici su larga scala (ShapeNet) e su immagini di oggetti del mondo reale (Pix3D).

In particolare, nel processo di addestramento, l’obiettivo era di generare una forma 3D in formato voxel a partire da modelli CAD 3D e, di conseguenza, è stato utilizzato un metodo di rendering per ciascun modello 3D basato sulla creazione di immagini con vista casuale e sfondo uniforme.

3.2 Pix2Vox

Per questa ricerca è stato utilizzato come riferimento il modello di rete neurale “*Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images*” [43]. Pix2Vox è abile nel ricostruire in maniera raffinata la forma 3D di un oggetto, la quale è rappresentata da una griglia voxel 3D dove 0 rappresenta una cella vuota e 1 denota una cella occupata. La ricostruzione 3D avviene sia a partire da una singola immagine RGB (*Single-view Reconstruction*) che da immagini multiple (*Multi-view Reconstruction*).

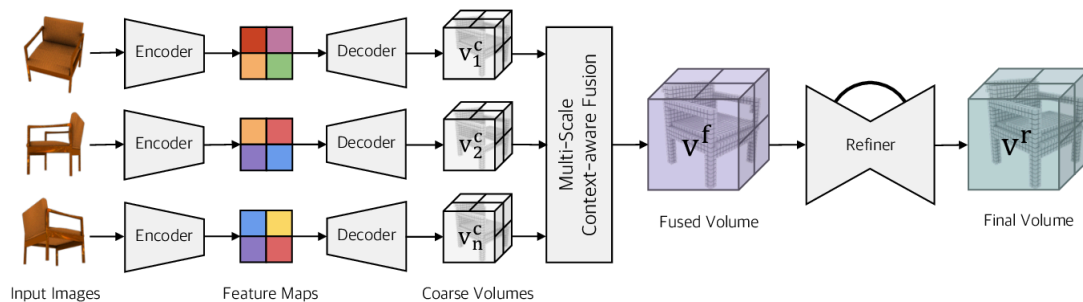


Figura 3.7: Una visione della rete proposta Pix2Vox.

Come presentato in figura 3.7, il modello Pix2Vox si compone di quattro moduli:

1. L'**Encoder** produce mappe di caratteristiche (feature map) delle immagini in input.
2. Il **Decoder**, per ogni mappa, genera il corrispondente volume grossolano 3D.
3. Il **Context-aware fusion module** seleziona, per ogni parte dei volumi grossolani, in maniera additiva tramite delle mappe-punteggio (score map) le ricostruzioni di elevata qualità generando un volume 3D fuso sfruttando tutte le informazioni delle immagini in input senza perdita di memoria.
4. Il **Refiner** corregge le parti mancanti del modello fuso 3D per ottenere una ricostruzione raffinata.

Tuttavia, per aver un buon bilanciamento tra precisione e dimensioni del modello, sono state implementate due versioni del framework proposto: Pix2Vox-F e Pix2Vox-A (figura 3.8).

In sintesi, l'architettura di Pix2Vox si concentra sull'elaborazione di immagini bidimensionali e sulla generazione di modelli 3D corrispondenti. Le reti neurali convoluzionali (CNN) sono più adatte per questo tipo di compito, poiché sono progettate per l'estrazione di feature da immagini e la convoluzione spaziale è particolarmente efficace nel rilevare pattern visivi nelle immagini.

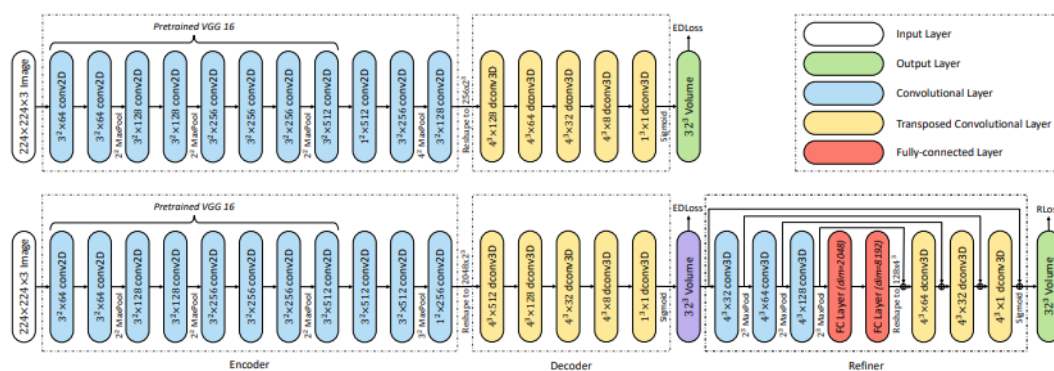


Figura 3.8: Architettura della rete Pix2Vox-F (alto) e Pix2Vox-A (sotto).

3.2.1 Fase di training

L'addestramento o *training* rappresenta il processo cruciale per ogni rete neurale, durante il quale la rete neurale impara dai dati e regola i suoi parametri interni per eseguire il compito desiderato. In particolare, ogni rete apprende dalle relazioni nei dati di addestramento, acquisisce delle caratteristiche rilevanti e migliora le proprie prestazioni attraverso l'ottimizzazione dei suoi parametri. Nel caso di reti dedicate alla ricostruzione di modelli 3D, i dataset di addestramento contengono solitamente coppie di dati contenenti le immagini bidimensionali e i corrispondenti modelli 3D dell'oggetto rappresentato.

Durante una sessione di training la rete neurale analizza l'intero dataset usato

per l'addestramento, costituito da input e output noti, per un determinato numero di volte predefinito o arbitrario, detto "epoch". Ogni epoca rappresenta un ciclo completo attraverso il dataset durante la quale la rete elabora tutti i dati nel dataset allo scopo di apprendere e migliorare le proprie capacità.

Al termine di ciascuna epoca viene calcolato un errore di addestramento, noto come "training error", che misura la differenza tra l'output prodotto dalla rete e l'output atteso. Nel nostro caso, il training error riflette quanto accuratamente il modello ricostruito sia più fedele al modello originale.

L'addestramento può essere terminato quando viene raggiunto un numero predefinito di epoche o quando l'errore di addestramento scende al di sotto di una soglia prefissata, spesso stabilita a un livello molto basso.

Alla fine di una sessione di addestramento, la rete neurale avrà ottimizzato i suoi parametri interni, detti "pesi", che sono responsabili delle interconnessioni tra i vari strati della rete. Questi verranno poi estratti e salvati, evitando la necessità di addestrare nuovamente l'intera rete da zero. Una volta completato il processo di addestramento, la rete sarà pronta per eseguire la sua funzione specifica, come la ricostruzione di modelli 3D da immagini, con prestazioni migliori rispetto all'inizio del processo di addestramento.

Tuttavia, trovare un equilibrio tra la durata dell'addestramento e la qualità dei risultati è importante. Alcuni principali fattori che influenzano la durata e il consumo di risorse durante l'addestramento sono:

- Dimensione del dataset: La dimensione del dataset di addestramento è un fattore chiave. Un dataset più grande richiederà più tempo per l'addestramento poiché la rete deve analizzare più dati.
- Complessità dell'architettura: L'architettura della rete, compresi il numero di strati, i nodi e le connessioni, influenzerà notevolmente la velocità dell'addestramento.
- La dimensione del batch, cioè il numero di campioni utilizzati in ciascuna iterazione durante il processo di addestramento, ha un grande impatto. Batch più grandi possono accelerare l'addestramento ma richiedono più risorse computazionali.
- Numero di epoche: Il numero di epoche, cioè il numero di volte che l'intero dataset viene utilizzato durante l'addestramento, influenza direttamente la durata totale. Un numero troppo elevato di epoche può portare a un addestramento eccessivo e quindi al fenomeno di *overfitting*.

Tuttavia, per quanto riguarda la rete Pix2Vox, durante il processo di training vengono coinvolti i quattro moduli dell'architettura mostrati in figura 3.1.

Encoder. L'encoder, composto dai primi nove strati convoluzionali di VGG16 pre-allenata e da strati aggiuntivi per l'incorporamento delle informazioni semantiche, viene addestrato per estrarre rappresentazioni significative dall'immagine di input. L'obiettivo è catturare le caratteristiche importanti dell'immagine che saranno utili per la ricostruzione 3D.

Decoder. Il decoder, costituito da cinque strati convoluzionali tridimensionali trasposti, è addestrato per generare volumi 3D a partire dalle rappresentazioni estratte dall'encoder. In questa fase, il decoder impara a trasformare le mappe di caratteristiche 2D in volumi 3D che rappresentano la forma dell'oggetto. L'obiettivo è produrre volumi 3D coerenti con l'oggetto rappresentato nell'immagine di input.

Context-aware Fusion Module. Il modulo di fusione Context-aware è addestrato per selezionare in modo adattivo e fondere le ricostruzioni di alta qualità da diverse prospettive. Le mappe di punteggio, mostrate in figura 3.9, vengono generate per ciascun volume approssimativo e vengono fuse attraverso una somma pesata secondo i punteggi. L'obiettivo è ottenere una fusione ottimale delle diverse ricostruzioni per generare un volume 3D completo e accurato dell'oggetto.

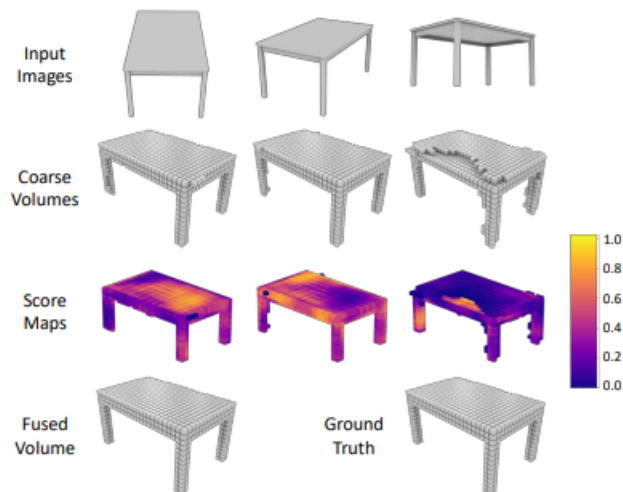


Figura 3.9: Illustrazione funzionamento delle mappe di punteggio.

Refiner. Il rifinitore, che funge da rete residua, è addestrato per correggere le parti erroneamente recuperate dei volumi 3D. Segue una struttura di codificatore-decodificatore 3D con connessioni U-net, grazie le quali la struttura locale nel volume fuso può essere preservata, ed in particolare è composto da tre strati convoluzionali 3D, ognuno dei quali ha un set di filtri. Attraverso il processo di addestramento, il rifinitore impara a raffinare le predizioni del decoder, migliorando la qualità complessiva della ricostruzione 3D.

Nello specifico, per la fase di training vengono utilizzate come input immagini RGB bidimensionali di dimensioni 224x224px e sia Pix2Vox-F che Pix2Vox-A sono addestrati utilizzando l'ottimizzatore ADAM⁸. Il tasso di apprendimento iniziale è impostato su 0.001 e viene ridotto del 50% dopo 150 epoche. Inizialmente, entrambe le reti vengono addestrate, ad eccezione del modulo di fusione Context-aware, usando immagini a singola vista per 250 epoche; successivamente, l'intera rete viene addestrata utilizzando un numero casuale di immagini di input per 100 epoche.

3.2.2 Ricostruzione degli oggetti 3D

Dopo aver appreso a mappare le caratteristiche dell'immagine 2D alle corrispondenti rappresentazioni 3D, in fase di addestramento, nel processo di ricostruzione la rete applica il "processo inverso" cercando di generare una rappresentazione 3D dell'oggetto che sia coerente con l'immagine fornita.

La coerenza tra le immagini di input per la ricostruzione e quelle utilizzate nel training è fondamentale, in quanto se la rete è stata addestrata su immagini di una certa dimensione e tipo di canali, avrà imparato a lavorare con quelle caratteristiche specifiche.

Dunque, per avere una fedele e accurata ricostruzione dei modelli 3D, è necessario mantenere la stessa risoluzione e lo stesso numero di canali delle immagini durante la fase di ricostruzione.

In particolare, la fase di ricostruzione nelle reti Pix2Vox-F e Pix2Vox-A coinvolge

⁸L'ottimizzatore *Adaptive Moment Estimation* (ADAM) è un algoritmo di ottimizzazione molto diffuso utilizzato nell'addestramento di reti neurali profonde, noto per la sua efficacia e versatilità.

principalmente il processo in cui la rete genera una rappresentazione voxelizzata tridimensionale dell'oggetto a partire dalle immagini di input, come mostrato in figura 3.10.

Dopo aver appreso a mappare le caratteristiche dell'immagine 2D alle corrispondenti rappresentazioni 3D, in fase di addestramento, nel processo di ricostruzione la rete applica il "processo inverso" cercando di generare una rappresentazione 3D dell'oggetto che sia coerente con l'immagine fornita.

La coerenza tra le immagini di input per la ricostruzione e quelle utilizzate nel training è fondamentale, in quanto se la rete è stata addestrata su immagini di una certa dimensione e tipo di canali, avrà imparato a lavorare con quelle caratteristiche specifiche.

Dunque, per avere una fedele e accurata ricostruzione dei modelli 3D, è necessario mantenere la stessa risoluzione e lo stesso numero di canali delle immagini durante la fase di ricostruzione.

In particolare, la fase di ricostruzione nelle reti Pix2Vox-F e Pix2Vox-A coinvolge principalmente il processo in cui la rete genera una rappresentazione voxelizzata tridimensionale dell'oggetto a partire dalle immagini di input, come mostrato in figura 3.10.

Dopo aver appreso a mappare le caratteristiche dell'immagine 2D alle corrispondenti rappresentazioni 3D, in fase di addestramento, nel processo di ricostruzione la rete applica il "processo inverso" cercando di generare una rappresentazione 3D dell'oggetto che sia coerente con l'immagine fornita.

La coerenza tra le immagini di input per la ricostruzione e quelle utilizzate nel training è fondamentale, in quanto se la rete è stata addestrata su immagini di una certa dimensione e tipo di canali, avrà imparato a lavorare con quelle caratteristiche specifiche.

Dunque, per avere una fedele e accurata ricostruzione dei modelli 3D, è necessario mantenere la stessa risoluzione e lo stesso numero di canali delle immagini durante la fase di ricostruzione.

In particolare, la fase di ricostruzione nelle reti Pix2Vox-F e Pix2Vox-A coinvolge principalmente il processo in cui la rete genera una rappresentazione voxelizzata tridimensionale dell'oggetto a partire dalle immagini di input, come mostrato in figura 3.10.



Figura 3.10: A partire dall'immagine in input, ricostruzione in formato voxel rispetto al modello di riferimento.

3.3 Pix3D

Il dataset di immagini di riferimento per questa ricerca, in fase di ricostruzione dei modelli 3D, è stato *Pix3D* [35]. Questa scelta è stata effettuata in relazione al fatto che molti dataset esistenti contengono solo immagini sintetiche e non presentano un allineamento tra l'immagine 2D e la corrispondente forma 3D. Di conseguenza, Pix3D rappresenta la soluzione migliore in quanto è un dataset su larga scala contenente immagini reali dell'oggetto e il corrispondente modello 3D, usato come “*ground truth*”.

Nell'ambito della Computer Vision e nel campo 3D, è fondamentale avere dei dati che riflettano una situazione reale. È altrettanto importante avere immagini che mostrino tali modelli in contesti realistici, e di conseguenza è necessario avere dei dataset che offrono una corrispondenza tra immagine e modello 3D.

Purtroppo molti dataset ad oggi disponibili hanno delle limitazioni. Esistono grandi set di dati contenenti modelli 3D ma non dotati di immagini reali. Altri contengono invece immagini reali ma con un allineamento approssimativo tra immagini e forme 3D, poiché gli oggetti nelle immagini sono abbinati a un set predefinito di modelli 3D e non al loro set reale. Mentre quelli che combinano immagini reali e modelli 3D sono di piccole dimensioni.

In sintesi, la necessità di avere un dataset come Pix3D deriva dalla crescente richiesta di algoritmi di visione artificiale che possano comprendere e analizzare oggetti 3D in immagini del mondo reale.

Pix3D ha 395 modelli 3D corrispondenti a 9 categorie di oggetti come mobili, utensili, veicoli e oggetti vari, e per ogni modello presenta una serie di immagini del relativo oggetto reale ripreso in diversi contesti (figura 3.11).

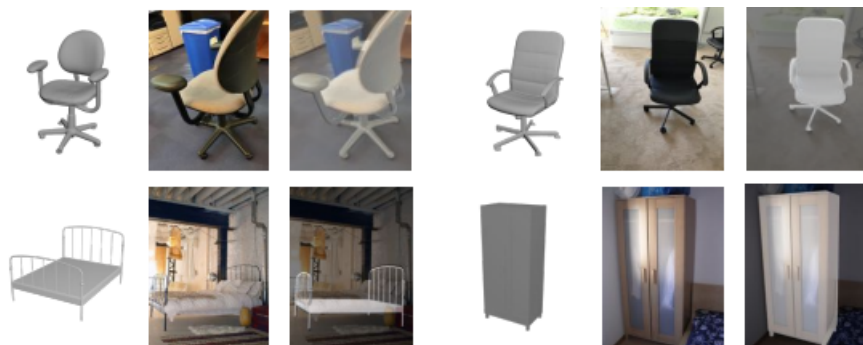


Figura 3.11: Esempi di immagini e forme in Pix3D. Da sinistra a destra: forme 3D, immagini 2D e allineamento immagine-forma.

3.4 Linguaggio Python

Tutta la componente di scripting e, in generale, l'intera parte legata alla programmazione è stata implementata utilizzando il linguaggio di programmazione

Python [25].

Python è un linguaggio di programmazione multi-paradigma di alto livello che supporta più paradigmi di programmazione, tra cui la programmazione procedurale, orientata agli oggetti e funzionale. È un linguaggio interpretato, il cui codice viene eseguito direttamente da un interprete semplificando il processo di sviluppo e debugging. Inoltre, dispone di una vasta libreria standard con relativi moduli che consentono di eseguire attività specifiche. Molti sono i vantaggi:

- La sintassi e la struttura sono semplici da imparare e da usare rispetto ad altri linguaggi di programmazione come *C* e *Java*.
- Linguaggio multiplatforma, permettendo così di poter girare su diversi sistemi operativi come *Linux*, *MacOS*, *Windows* e *Unix*.
- Estremamente versatile.
- Possibilità di accedere ad un'ampia libreria standard contenente funzioni di calcolo scientifico e codici riutilizzabili per quasi tutte le attività.
- Semplifica le attività, aiutando gli sviluppatori a scrivere solo poche righe di codice.
- Portabilità, ossia una sorgente scritta su una piattaforma potrà essere successivamente interpretata su un sistema operativo differente.
- Presenza di una robusta community mondiale per poter ricevere supporto e risolvere eventuali problemi.

Per lo sviluppo e l'implementazione della rete neurale sono state impiegate le principali librerie e framework:

Numpy: Numpy [18] è la libreria fondamentale per il calcolo scientifico.

Matplotlib: Matplotlib [14], usata in combinazione con la libreria Numpy, è una libreria per la visualizzazione dei dati. È utilizzata per creare grafici e tracciati in diversi formati e rendere comprensibili dati più complessi. Tale libreria offre flessibilità e opzioni per personalizzare l'aspetto dei grafici, l'aggiunta di etichette, titoli, legende e molto altro.

PyTorch: PyTorch [26] è un framework automatico utilizzato per applicazioni come la Computer Vision.

TensorFlow: TensorFlow [36] è un framework open-source che fornisce strumenti per costruire e addestrare reti neurali e modelli di machine learning in modo efficiente, supportando l'addestramento sia su CPU che su GPU. È progettato per lavorare con tensori, ovvero strutture multidimensionali simili a array o matrici.

TensorFlow permette una maggiore ottimizzazione nell'addestramento grazie all'utilizzo di un modello basato su grafici computazionali. TensorBoard è uno strumento di visualizzazione integrato in TensorFlow che consente di monitorare l'addestramento e l'analisi dei modelli visualizzando grafici, statistiche.

In particolare, Pix2Vox è implementata in PyTorch.

4 Pipeline di lavoro e metriche di valutazione

4.1 Descrizione pipeline per la creazione del dataset

Dopo un'approfondita analisi del funzionamento delle reti neurali convoluzionali, con particolare attenzione alle fasi di addestramento e test, e proceduto con la selezione del caso studio per il nostro progetto di ricerca, successivamente ci si è focalizzati sull'adattare tale caso studio al dataset di immagini sintetiche precedentemente sviluppato nella tesi precedente [22]. Il dataset, impiegato durante la fase di addestramento del nostro modello neurale, è stato realizzato attraverso l'implementazione di una specifica pipeline di lavoro comprensivo di diverse fasi:

1. **Scelta delle categorie di oggetti.** Selezione delle categorie di oggetti in base alle esigenze specifiche del progetto e scelta dei modelli più adatti per il raggiungimento dello scopo stabilito.
2. **Raccolta di modelli 3D.** Raccolta attenta dei modelli 3D sia in termini di compatibilità col software utilizzato che sul tipo di formato. I modelli sono stati ricercati da fonti esterne, come siti web dedicati, o creati da zero, a seconda delle esigenze.
3. **Creazione/raccolta dei materiali.** Una varietà di materiali da applicare ai modelli sono stati ottenuti da dataset di texture disponibili online o creati manualmente.
4. **Sistemazione dei modelli e applicazione dei materiali.** Per facilitare il processo di gestione luci e camera, le dimensioni dei modelli sono state normalizzate e, in secondo luogo, sono stati applicati i materiali. L'applicazione dei materiali ai modelli può essere effettuata in vari modi, inclusi l'applicazione casuale o manuale dei materiali, a seconda delle esigenze specifiche di ciascun oggetto.
5. **Setup delle luci.** Un'illuminazione adeguata della scena prevede la scelta del tipo di luci, del colore e della potenza luminosa, legati dallo specifico utilizzo del dataset. Le luci sono state posizionate in modo da illuminare al meglio il modello 3D evitando effetti indesiderati come ombre nette o riflessioni indesiderate.
6. **Setup della camera.** Definite le luci, sono stati settati i parametri della telecamera, come la lunghezza focale, la profondità di campo e l'apertura del diaframma, in base alle esigenze specifiche del progetto. Qualora fosse sta-

to necessario acquisire l'oggetto da varie prospettive, venivano configurati keyframe per definire la posizione della telecamera.

7. **Rendering.** Durante la fase di rendering, si è scelto se produrre immagini a colori o in scala di grigi e se ottenere mappe di profondità o mappe normali relative al modello, per poter ottenere così dei risultati fotorealistici.

In particolare, per gestire ciascuna fase del lavoro, è stato possibile creare script specifici da eseguire in Blender oppure, come è stato fatto nella ricerca precedente, sviluppare un programma dedicato in grado di automatizzare completamente l'intero processo. Grazie alle funzionalità offerte sia da Blender che da Blender-Proc, è stato così possibile gestire la creazione dei dataset utilizzati per le diverse prove.

4.2 Feature per il training

Dopo aver condotto un'analisi dettagliata delle reti neurali convoluzionali e selezionato il caso studio, il passo successivo è stato valutare l'importanza dei parametri chiave dell'addestramento, nello specifico, della rete neurale Pix2Vox utilizzando il dataset di immagini sintetiche, a partire da modelli 3D, creato precedentemente.

In particolare, si ci si è concentrati principalmente sulla dimensione delle immagini, materiali applicati ai modelli, colore delle immagini e illuminazione.

4.2.1 Dimensione dell'immagine

L'effetto della dimensione dell'immagine nel deep learning è un aspetto importante da considerare durante la progettazione e l'addestramento di reti neurali. Quest'ultima influisce sulla complessità del modello, sulla richiesta di risorse computazionali, sulla quantità di dati necessaria, e su altri aspetti chiave. Dimensioni più grandi richiedono modelli più complessi e maggiore potenza di calcolo e, di conseguenza, tempi di addestramento più lunghi.

Ad esempio, una rete convoluzione (CNN) progettata per elaborare immagini ad

alta risoluzione, 1024x1024px, cattura maggiormente i dettagli fini di un'immagine ma, allo stesso tempo, richiede un numero maggiore di parametri rispetto ad una progettata per immagini a minore risoluzione, come 128x128px.

Tuttavia, analizzando i diversi studi relativi alla dimensione delle immagini utilizzate durante l'addestramento, si è notato che molti di questi si concentrano principalmente sul riconoscimento di elementi all'interno delle immagini, piuttosto che sul contesto della ricostruzione 3D [29][31].

Pertanto, si è deciso di condurre diversi esperimenti utilizzando immagini a varie risoluzioni.



Figura 4.1: Percezione dei dettagli in base alla diversa risoluzione dell'immagine.

4.2.2 Texture e materiali

Un ulteriore aspetto è la gestione dei materiali applicati ad un oggetto. La possibilità di gestire i materiali dipende in gran parte dal tipo di rete e dall'approccio utilizzato nell'addestramento. Ad esempio, materiali complessi, ovvero costituiti da diverse "mappe" come la normal, la roughness o la specular map (figura 4.2), incrementano il numero di risorse e i tempi di rendering ma contribuiscono nel conferire un aspetto più realistico ai modelli. Di conseguenza, si sono effettuati diversi esperimenti per comprendere l'influenza dell'utilizzo di un modello con texture o senza materiale, durante la fase di training.

Inoltre, importante è anche la scelta riguardo immagini a colori piuttosto che in scala di grigi. Le immagini a colori sono generalmente più grandi in termini di dimensioni dei dati rispetto alle immagini in scala di grigi, a causa del numero di

canali. Ciò può incidere sulla memoria e capacità di calcolo necessarie per l'addestramento e l'elaborazione dei dati. Pertanto, lavorare con immagini in scala di grigi può ridurre notevolmente la complessità computazionale.

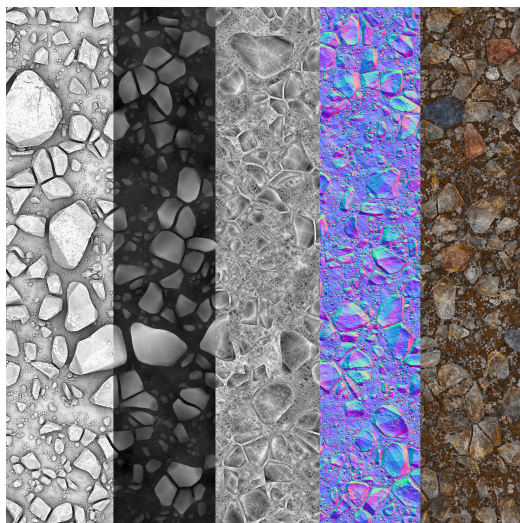


Figura 4.2: Differenti mappe di texture.

4.2.3 Illuminazione

L'illuminazione del modello 3D in fase di rendering è un aspetto importante da considerare quando si lavora con reti neurali per la ricostruzione 3D o altre applicazioni di Computer Vision. La presenza di ombre e riflessioni può influenzare il training della rete neurale e il risultato della ricostruzione in vari modi. Ad esempio, nel caso in cui la scena utilizzata contiene ombre o riflessioni intense, la rete potrebbe considerarle come parte integrante degli oggetti e portare così ad una ricostruzione distorta o inaccurata degli oggetti. Essendo, quindi, anche l'illuminazione un parametro influente, sono stati effettuati esperimenti in cui il modello 3D viene illuminato sia con una luce omogenea che col sistema di luce three-point lighting (figura 4.3).

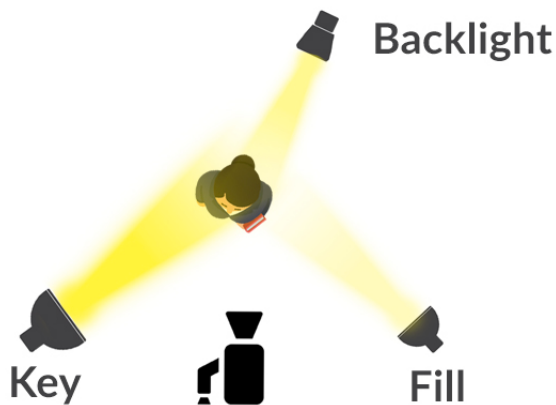


Figura 4.3: Sistema three-point lighting.

4.3 Chamfer Distance

Dopo aver individuato i parametri potenzialmente influenti nell'addestramento di una rete neurale e nella ricostruzione di oggetti tridimensionali, si è stabilito un metodo efficace per valutare le prestazioni della rete in termini di precisione nella ricostruzione dei modelli 3D.

La metrica di valutazione scelta per misurare l'accuratezza della ricostruzione del modello 3D rispetto all'originale è stata la *Chamfer Distance* [41]. L'utilizzo della Chamfer Distance è stato fondamentale per effettuare un confronto tra i risultati ottenuti nei nostri esperimenti, con la rete neurale Pix2Vox, e quelli ottenuti nella tesi precedente. Questo confronto ha permesso di analizzare l'andamento della metrica alle variazioni dei parametri chiave che influenzano il processo di addestramento, nello specifico nel contesto della ricerca attuale.

La Chamfer Distance (CD) è una metrica di valutazione utilizzata in molte applicazioni, tra cui la ricostruzione 3D, il riconoscimento di oggetti e la segmentazione di immagini. In particolare, è una metrica molto efficace per valutare la qualità di un modello rispetto a un modello di riferimento noto, o *ground truth*, quando si lavora con dati che possono essere sotto forma di punti 3D o nuvole di punti.



Figura 4.4: Rappresentazione grafica della Chamfer Distance.

Considerando due insiemi differenti di nuvole di punti, un insieme previsto e un insieme di riferimento, per ogni punto di questi insiemi, si calcola la distanza al punto più vicino nell'insieme di punti di riferimento e la distanza al punto più vicino nell'insieme previsto (figura 4.4). La media delle distanze al quadrato tra i punti previsti e i punti di riferimento è chiamata Chamfer Distance. In figura 4.5 viene mostrata la formula matematica.

$$d_{CD}(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|y - x\|_2^2$$

Figura 4.5: Formula matematica per la Chamfer Distance.

Tuttavia, questa metrica è particolarmente utile quando si lavora con dati come nuvole di punti 3D, tenendo conto delle discrepanze tra i punti delle due nuvole di punti e fornendo una valutazione della somiglianza o della dissimilarità tra i due insiemi di punti.

4.4 Altre metriche di valutazione

In generale, le metriche di valutazione sono strumenti essenziali nell'apprendimento automatico per misurare le prestazioni dei modelli e degli algoritmi in base agli obiettivi specifici del problema. Queste metriche quantificano l'errore o la discrepanza tra le previsioni del modello e i dati di riferimento.

Oltre la Chamfer Distance, esistono diverse metriche adatte a differenti tipi di problemi, e la loro scelta influisce sulla valutazione e l'ottimizzazione dei modelli. Alcune di queste sono:

- **IoU**

L'*Intersection over Union* (IoU) [28] è utilizzata come metrica principale per valutare la qualità di un algoritmo o un modello nell'identificare oggetti o aree specifiche nell'immagine. È una misura di sovrapposizione tra due regioni o maschere, spesso utilizzata per confrontare la similarità tra una regione prevista da un modello con una regione di riferimento.

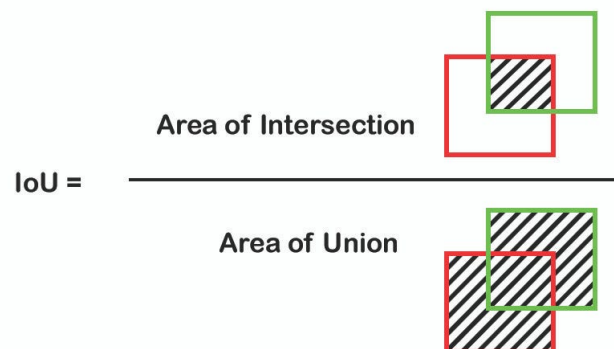
$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$


Figura 4.6: Formula matematica della metrica IoU.

In figura 4.6 viene mostrata la formula matematica, dove l'area dell'Intersezione rappresenta la superficie comune tra la regione prevista e il ground truth, mentre l'area dell'Unione rappresenta l'area totale coperta sia dalla regione prevista che dal ground truth. Il risultato sarà un valore tra 1 e 0, in cui il valore massimo 1 indica una sovrapposizione perfetta tra la previsione e il ground truth, ovvero le due regioni sono identiche. Il valore 0

indica che la previsione e il ground truth non hanno alcuna sovrapposizione.

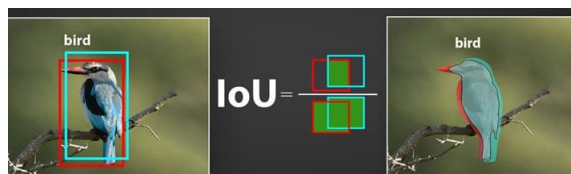


Figura 4.7: Rappresentazione del compito di rilevamento dell'oggetto.

Un esempio pratico viene mostrato in figura 4.7. L'obiettivo è quello di migliorare costantemente la previsione fino a quando il box blu e il box verde si sovrappongono perfettamente. Quanto più l'IoU si avvicina a 1, tanto migliore sarà la previsione del modello.

- **EMD**

L'*Earth Mover's Distance* (EMD) [39] è una metrica di valutazione utilizzata per misurare la discrepanza tra due modelli o previsioni di ricostruzione 3D. Spesso lavora con distribuzioni rappresentate da punti chiave estratti da oggetti o scene 3D, o intere nuvole di punti che descrivono la geometria di un oggetto o di una scena. Un basso valore di EMD indica che le due distribuzioni di punti 3D sono molto simili in termini di posizione e distribuzione spaziale, mentre un valore più alto indica una maggiore discrepanza tra le previsioni del modello e i dati di riferimento.

- **MSD**

La *Metric Surface Distance* (MSD) [1] è una metrica utilizzata per valutare quanto bene una superficie tridimensionale generata da un modello corrisponda a una superficie di riferimento in applicazioni di ricostruzione 3D e segmentazione.

Il calcolo della MSD coinvolge solitamente un comparazione punto per punto tra le due superfici, per valutare l'allineamento tra il modello di ricostruzione 3D e la superficie di riferimento. Un valore basso indica una maggiore precisione nella corrispondenza tra modello e ground truth.

5 Progettazione e realizzazione del lavoro

Dopo aver condotto un'approfondito studio della letteratura esistente, analizzato le diverse reti neurali dedicate alla ricostruzione 3D Single-view, definito il caso studio e il dataset di immagini da utilizzare per l'addestramento, delineato gli obiettivi da conseguire e la metrica adatta per valutare le prestazioni dei modelli ricostruiti, si è passati alla realizzazione pratica delle attività di ricerca svolta in questa tesi.

5.1 Rete Neurale

Il primo compito ha riguardato l'implementazione della rete neurale Pix2Vox [43]. Questa rete è progettata per operare su due tipi di immagini: quelle sintetiche provenienti dal dataset ShapeNet e quelle reali tratte dal dataset Pix3D. Per ciascuno di questi dataset, l'autore ha fornito le diverse immagini insieme ai corrispettivi modelli espressi nel formato voxel. Tuttavia, è importante ricordare che sono state sviluppate due versioni del modello Pix2Vox, denominate Pix2Vox-A e Pix2Vox-F, le quali sono state preaddestrate utilizzando il dataset ShapeNet.

Data la notevole richiesta di risorse di calcolo durante la fase di addestramento, la rete Pix2Vox è stata impiegata su una workstation equipaggiata con una GPU Nvidia GeForce 1060 da 6GB di memoria dedicata, necessaria per eseguire le diverse elaborazioni richieste.

5.1.1 Implementazione della rete neurale

L'implementazione della rete ha coinvolto due fasi principali: inizialmente, è stato necessario installare i requisiti specifici forniti dall'autore e configurare i parametri adeguatamente. I parametri fondamentali per far sì che la rete neurale Pix2Vox funzioni adeguatamente sono:

- **Pesi dell'encoder.** I *pesi* rappresentano i parametri fondamentali che la rete neurale utilizza per apprendere e fare previsioni sui dati di input. Durante il processo di addestramento, la rete neurale riceve dei dati, li elabora attraverso una serie di strati interconnessi e confronta le sue previsioni con

gli obiettivi desiderati. Gli errori che vengono calcolati sono utilizzati per regolare i pesi, i quali vengono inizializzati in modo casuale. Successivamente, vengono ottimizzati attraverso iterazioni successive per minimizzare l'errore tra le previsioni della rete e i dati di addestramento.

Nella fase di ricostruzione, questi parametri sono essenziali in quanto codificano le conoscenze apprese dalla rete durante l'addestramento, consentendo alla rete di riconoscere pattern e fare previsioni accurate.

In genere, i pesi vengono salvati periodicamente (ogni 10 o 100 epoch) in file o modelli specifici, in modo che possano funzionare come backup in caso di guasti del sistema ed essere riutilizzati quando si desidera, senza dover ripetere l'intero processo di addestramento.

- **Device GPU.** Quando si lavora con reti neurali implementate utilizzando il framework PyTorch, è possibile selezionare la risorsa hardware da utilizzare per eseguire i calcoli, come la CPU o la GPU. La scelta tra queste due risorse può avere un impatto significativo sulle prestazioni dell'addestramento e sull'efficienza computazionale complessiva del modello. Nel caso della rete Pix2Vox si è scelto di utilizzare la GPU, così da poter ridurre notevolmente i tempi di addestramento.
- **Img_W e Img_H.** Le dimensioni delle immagini, *img_W* (larghezza) e *img_H* (altezza), sono parametri importanti poiché rappresentano nello specifico le dimensioni in pixel di quest'ultime che la rete riceve come dati di input. È importante notare che le dimensioni dell'immagine di input possono influenzare la complessità e le prestazioni del modello. Immagini più grandi possono richiedere più risorse computazionali, ma possono contenere più dettagli. Nello specifico, la rete Pix2Vox gestisce di base immagini con dimensione 224x224px.
- **Crop_img_W e Crop_img_H.** La dimensione di "crop" dell'immagine, *crop_img_W* (larghezza del ritaglio) e *crop_img_H* (altezza del ritaglio), si riferisce alla porzione dell'immagine originale che viene ritagliata prima di essere passata alla rete neurale. Questo processo di ritaglio, casuale o sistematico, è spesso utilizzato per standardizzare le dimensioni delle immagini di input quando quest'ultime sono di dimensioni diverse.
- **Num_vox.** Il numero di *voxel* si riferisce alla quantità di elementi di volume tridimensionali all'interno di uno spazio o di un'immagine tridimensionale. Un voxel è un cubo tridimensionale che rappresenta una piccola porzione di uno spazio 3D, e di conseguenza il numero di voxel corrisponde al totale di questi cubi all'interno di una griglia 3D. Nello specifico, questo parametro definisce la risoluzione spaziale dell'ambiente tridimensionale,

ovvero quanto dettagliata sarà la rappresentazione volumetrica dello spazio 3D.

In questo caso, il parametro `num_vox`, originariamente settato a 32 dall'autore di Pix2Vox, è rimasto inalterato.

- **Batch_size.** Nel momento in cui si lavora con grandi dataset, per far sì che questi possano essere elaborati tutti in una volta, vengono suddivisi in sottogruppi più gestibili chiamati batch. La *batch_size* si riferisce alla dimensione dei sottogruppi, ovvero al numero di campioni che si propagano attraverso la rete neurale. La scelta del batch size influisce sul numero di iterazioni necessarie per completare un'epoca, ossia per attraversare l'intero dataset una volta. In particolare, il numero di iterazioni necessarie si ottiene dividendo il numero totale di immagini nel dataset per il batch size.

Per la scelta della batch size non vi è uno standard da eseguire, ma può variare a seconda delle dimensioni del set di dati, della complessità del modello e delle risorse hardware disponibili. Batch size tipici sono 32, 64 o 128, solitamente potenze di 2. Una batch size più grande può portare a una maggiore velocità di addestramento, ma richiede più memoria del sistema; al contrario, una batch size più piccola può richiedere meno memoria, ma l'addestramento può richiedere più tempo a causa del maggior numero di iterazioni necessarie per attraversare l'intero set di dati di addestramento.

Nel nostro caso il modello Pix2Vox utilizza una batch size pari a 64, ma quando si lavora con dati 3D o immagini ad alta risoluzione, i requisiti di memoria possono aumentare significativamente, quindi è essenziale adattare il batch size alle capacità della GPU disponibile. Per tale motivo si è deciso di impostare questo parametro a 16.

- **Learning_rate.** Il tasso di apprendimento, o *learning rate*, determina la velocità della rete neurale nell'aggiornare i pesi durante l'addestramento. Un learning rate troppo alto può causare problemi come l'instabilità o la divergenza e trascurare i dettagli, mentre un learning rate troppo basso può rallentare l'addestramento. Il learning rate deve essere sintonizzato attentamente per adattarsi al problema specifico della rete neurale e poter ottenere una convergenza stabile ed efficiente durante l'ottimizzazione. Il valore di learning rate proposto dall'autore di Pix2Vox è stato 0,0001.

Dopo aver analizzato i parametri cruciali per il funzionamento della rete, si è proceduto con la fase di ottimizzazione. Poiché un aumento della dimensione

di batch comportava una riduzione delle prestazioni, si è scelto di ridurre la dimensione del batch a 16. Ciò è stato fatto per determinare se questa modifica potesse effettivamente accelerare le prestazioni della rete. Inoltre, per caricare e pre-elaborare i dati durante l'addestramento si è utilizzato un certo numero di *worker*, ovvero un numero di processi paralleli o thread utilizzati. Inizialmente settato a 4, si è deciso di aumentare quest'ultimo ad 8 per accelerare il processo di addestramento e consentire un caricamento più rapido dei dati. Tuttavia, è importante bilanciare questo numero in base alla capacità del sistema e alla quantità di dati disponibili in quanto un uso eccessivo potrebbe causare un sovraccarico e ridurre le prestazioni.

Prima di avviare la fase di addestramento effettiva, utilizzando il nostro dataset di riferimento, e successivamente condurre gli esperimenti, si è deciso di condurre una valutazione preliminare del funzionamento della nostra rete e identificare eventuali criticità. A tal fine, abbiamo impiegato il dataset Shapenet fornito dall'autore ed eseguito una fase di test utilizzando i pesi ottenuti durante la fase di preaddestramento del modello Pix2Vox-F.

5.2 Addestramento della rete

Dopo aver convalidato la corretta esecuzione delle fasi di addestramento e test utilizzando il dataset ShapeNet [7], composto da immagini con dimensioni di 224x224px, si è proceduti all'addestramento della rete adottando il dataset precedentemente creato dall'ingegnere Piparo nella tesi [22]. Tale dataset di riferimento è stato generato, a partire da modelli 3D di tre diverse categorie quali sedie, tavoli e armadi, attraverso un processo di rendering utilizzando il software Blender e, in particolare, sfruttando la funzione "render" di BlenderProc. Tuttavia, questo dataset oltre a includere le immagini sintetiche delle tre categorie, comprende anche i modelli corrispondenti espressi in forma voxel, ottenuti grazie al tool Binvox [2].

In particolare, per l'addestramento di Pix2Vox è stata selezionata la categoria sedie.

5.2.1 Tassonomia

I modelli Pix2Vox-A e Pix2Vox-F sono stati preaddestrati utilizzando il dataset ShapeNet, il quale contiene modelli 3D di diverse categorie semantiche organizzate secondo una *tassonomia*⁹ gerarchica definita. La tassonomia di ShapeNet organizza i modelli 3D in categorie semantiche gerarchiche secondo la struttura di WordNet [17]. Ad esempio, a un livello di basso livello, ci possono essere categorie come sedie, tavoli e armadi, mentre a un livello più alto ci potrebbero essere categorie più generiche come mobili. Questo approccio consente di raggruppare oggetti simili in categorie più ampie, permettendo di navigare all'interno del dataset in modo più efficiente.

Di conseguenza, per addestrare adeguatamente la rete Pix2Vox col nostro dataset, è stato necessario creare il file di tassonomia mancante.

Il file di tassonomia *json*, creato per ogni categoria di oggetti sedia, contiene il *taxonomy_id*, ovvero l'identificativo della categoria di oggetti, il *taxonomy_name*, ovvero il nome della categoria e la suddivisione dei vari modelli nelle fasi di addestramento, test e convalida.

Di seguito viene mostrato un esempio del file *json* per una specifica categoria sedia.

```
1 [{
2   "taxonomy_id": "512_chair_flat",
3   "taxonomy_name": "chair",
4   "train": [chair4, chair10, chair22, ...],
5   "test": [chair8, chair25, chair27, ...],
6   "val": [chair20, chair26, chair28, ...]
7 }]
```

La suddivisione del dataset in training, test e validation è una pratica comune dell'addestramento di reti neurali. Tuttavia, non esiste una regola specifica che sia universalmente applicabile per la suddivisione di un dataset in training, va-

⁹Il termine *tassonomia* viene utilizzato in vari campi per indicare la classificazione sistematica di oggetti, concetti o fenomeni in categorie o classi basate sulle loro caratteristiche comuni.

validation e test set (figura 5.1). Solitamente, la scelta delle proporzioni dipende dal numero di campioni presenti nel dataset.

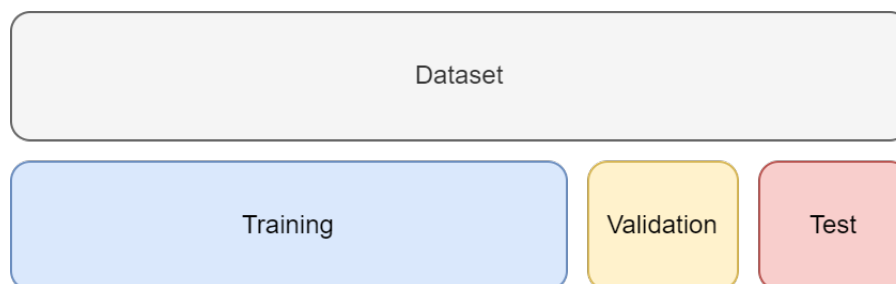


Figura 5.1: Suddivisione del dataset.

Training Set è la porzione fondamentale del dataset utilizzata per addestrare il modello di rete neurale, la quale impara a rappresentare i dati e ottimizza i suoi pesi e parametri in modo da minimizzare l'errore di training.

Validation Set è una porzione separata del dataset, non utilizzata durante l'addestramento, che serve per regolare l'architettura del modello e far sì che l'overfitting, ossia quando il modello si adatta troppo ai dati di training, non si verifichi.

Test Set è utilizzato per valutare le prestazioni della rete, addestrata ed ottimizzata, e aiuta a determinare la sua efficacia su nuovi dati.

Di solito, la maggior parte dei dati è assegnata al training set, tipicamente un intervallo dal 70% all'80% dei dati totali, mentre il rimanente 20-30% dei dati viene diviso in parti uguali tra il validation e test set.

Nel nostro caso, abbiamo allocato l'80% dei dati per la fase di addestramento e il restante 20% diviso tra le fasi di test e convalida.

5.2.2 Lettura file

Nel contesto della configurazione della rete, si è passati alla gestione dell'accesso ai dati del dataset di riferimento, denominato "Piparo". Questo dataset comprendeva un file di tassonomia, immagini sintetiche delle diverse categorie di sedie e i corrispondenti modelli voxel, tutti dati che sono stati organizzati in specifiche cartelle nel nostro sistema.

Per facilitare l'accesso ai dati, nel file di configurazione si è aggiunto del codice dedicato, specificando anche il percorso specifico per ciascun tipo di dato nel nostro dataset:

- **__C.DATASETS.PIPARO.TAXONOMY_FILE_PATH**: Questo percorso punta al file *json* che contiene le informazioni sulle tassonomie del dataset “Piparo”. Questo file è essenziale per comprendere la struttura e la categorizzazione delle sedie nel nostro dataset.
- **__C.DATASETS.PIPARO.RENDERING_PATH**: Questo percorso è utilizzato per le immagini di rendering in formato PNG. Il segnaposto '%s' è stato utilizzato per rappresentare dinamicamente il nome della tassonomia, il nome del campione e un numero che indica il materiale specifico delle sedie.
- **__C.DATASETS.PIPARO.VOXEL_PATH**: Questo percorso è progettato per i modelli voxel delle sedie nel formato *.binvox*. Anche qui, si utilizza il segnaposto '%s' per il nome del campione, consentendoci di accedere ai modelli voxel specifici associati a ciascuna sedia.

Inoltre, il dataset “Piparo” è diviso in una parte di addestramento (train) e una parte di test, per poter valutare e migliorare le prestazioni della rete neurale Pix2Vox nel riconoscimento e nella comprensione delle diverse categorie di sedie.

Per quanto riguarda la gestione dei file, sono state implementate due classi tramite script. La prima di esse, chiamata **‘PiparoDataset’**, è stata sviluppata per creare un dataset personalizzato per l'addestramento utilizzando il framework PyTorch. Questa classe include una serie di parametri fondamentali, tra cui il tipo di dataset, una lista di dizionari. Ciascun dizionario contiene dettagli sui file presenti nel dataset, comprese le informazioni sulle diverse categorie di sedie. Inoltre, è possibile specificare il numero di visualizzazioni delle immagini di rendering da selezionare per ciascun campione, insieme a trasformazioni opzionali da applicare a queste immagini.

Tuttavia, la rete non si limita a gestire solo le immagini di rendering ma considera anche i corrispondenti modelli voxel associati. Per tale motivo, carica le immagini di rendering mediante la libreria *OpenCV*, convertendole in array numpy di tipo *float32*, occupandosi anche del caricamento del volume dal file *binvox*, tramite il modulo *utils.binvox_rw*. Poiché questo modulo solitamente gestisce modelli *binvox* con dimensioni di 64, i volumi sono stati convertiti in array numpy di tipo *float32* utilizzando il metodo *ndarray.astype*.

La seconda classe, denominata **‘PiparoDataLoader’**, è responsabile del caricamento dei dati per il dataset “Piparo”. Essa contiene informazioni cruciali come una lista che racchiude dettagli sulle tassonomie del dataset, insieme a modelli di percorsi dei file per le immagini di rendering e i volumi corrispondenti. Questa classe svolge un ruolo fondamentale nel controllo dell’esistenza dei file e restituisce una lista completa dei campioni associati a ciascuna tassonomia, garantendo così che il processo di addestramento possa avvenire in modo efficace e senza problemi.

5.2.3 Ridimensionamento immagini

Dopo aver verificato la corretta lettura dei file e il funzionamento stabile della rete con questi dati, si è passati alla prima fase di addestramento, adattando il dataset alle specifiche richieste della rete Pix2Vox. Quest’ultima richiede immagini di dimensione 224x224px, mentre il nostro dataset originale conteneva immagini di dimensioni superiori, rispettivamente 512x512px e 1024x1024px. Perciò, tali immagini sono state ridimensionate alla dimensione desiderata di 224x224px. Nel contesto del training di reti neurali, è consuetudine che ogni rete neurale generi dei “pesi”, i quali rappresentano caratteristiche specifiche estratte dalle immagini di input. Questi valori solitamente vengono archiviati in file dedicati, al fine di utilizzarli nella fase successiva di ricostruzione.

Nel dettaglio, la rete Pix2Vox è stata sottoposta a un processo di addestramento lungo 1000 epoche, adottando un’operazione di salvataggio ogni 10 epoche per evitare perdite dei dati in caso di malfunzionamento o crash del sistema.

5.3 Ricostruzione 3D e calcolo della Chamfer Distance

Finita la fase di training e salvati automaticamente i pesi e lo stato del modello in apposite cartelle, si è proceduti alla fase di ricostruzione.

5.3.1 Voxel

Per assicurare il corretto funzionamento della rete durante la fase di test e per verificare il successo dell'addestramento della rete Pix2Vox, è stata implementata una procedura di verifica in tempo reale che coinvolgeva la visualizzazione dei modelli 3D in formato voxel. Grazie a ciò, venivano visualizzati sia i modelli generati dalla rete che i modelli di riferimento, o ground truth. Per la creazione dei modelli voxel è stato utilizzato il tool *Binvox* [2].

Binvox è uno strumento di conversione utilizzato per convertire i modelli tridimensionali, rappresentati come mesh, in una rappresentazione binaria tridimensionale, chiamata voxel grid. Questa voxel grid rappresenta l'oggetto tridimensionale in una griglia 3D di unità di volume chiamate voxel, ovvero dei punti equivalenti ai pixel in un'immagine 2D (figura 5.2). L'utilizzo di *Binvox* richiede, tramite terminale, che vengano specificati i percorsi dei file di input e output, insieme alla dimensione desiderata della griglia 3D e altri parametri di trasformazione. Questo tool è compatibile con i formati *.obj* e *.ply*, pertanto sono stati utilizzati modelli tridimensionali *.ply* ottenuti tramite uno script Python nella fase di ricerca precedent, per poi poterli convertire ulteriormente in formato voxel.

In particolare, è stata modificata la dimensione finale della griglia contenente il modello voxel. Mentre, nel contesto della rete neurale BSP-Net la dimensione della griglia era stata impostata a 64x64x64, si è deciso di cambiare e regolare tale parametro a 32x32x32, per soddisfare così i requisiti specifici della rete Pix2Vox. Inoltre, il formato di output scelto come "raw" è stato *.binvox*.

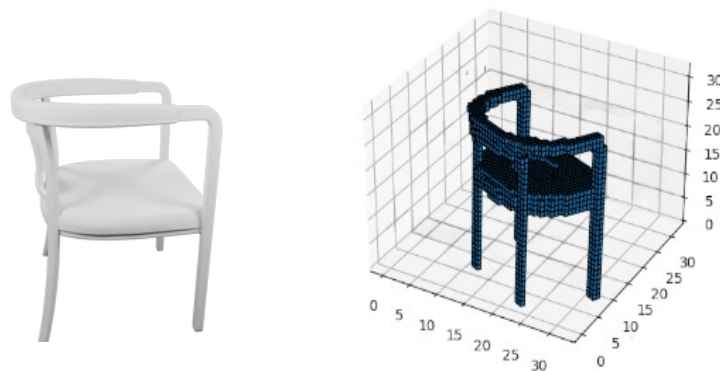


Figura 5.2: Modello in formato voxel generato col tool *Binvox*.

5.3.2 Calcolo Chamfer Distance

Per valutare con precisione l'accuratezza delle ricostruzioni tridimensionali, si è scelto di adottare l'utilizzo della metrica conosciuta come "Chamfer Distance" [41]. Questa metrica è specificamente progettata per valutare la somiglianza tra i modelli 3D ricostruiti e i modelli ground truth, entrambi espressi nel formato di nuvole di punti. È importante sottolineare una differenza significativa rispetto alla rete BSP-Net. Mentre quest'ultima aveva la capacità di ricostruire direttamente modelli 3D come nuvole di punti, facilitando il calcolo della metrica, dall'altra parte la rete Pix2Vox non ha questa stessa capacità intrinseca di generare direttamente nuvole di punti.

Di conseguenza, è stata adottata una soluzione indiretta integrando uno script Python precedentemente sviluppato dall'ingegnere Piparo, per calcolare la Chamfer Distance. In particolare, a livello di programmazione, poiché il calcolo della Chamfer Distance richiede degli array 2D con due parametri di input, quali il numero di punti e la dimensione dell'array, si è deciso di effettuare una manipolazione sul modello ricostruito e sul modello ground truth, rispettivamente **generated_volume** e **ground_truth_volume**, come mostrato di seguito.

```
1 gvl = generated_volume.reshape(-1, generated_volume.shape[-1])
2 gt = ground_truth_volume.(-1, ground_truth_volume.shape[-1])
3
4 sample_cd = chamfer_distance(gvl, gt)
```

Di base, i due volumi erano due array multidimensionali con dimensioni (1, 32, 32, 32), dove il valore 1 rappresenta la dimensione del batch, mentre gli altri tre assi corrispondono alle dimensioni spaziali del volume. Nello specifico, il metodo `.shape[-1]` è stato utilizzato per estrarre il valore 32, rappresentante la dimensione del volume lungo l'ultimo asse, mentre il parametro -1 del metodo `.reshape`, ha consentito una ristrutturazione dell'array in modo da ottenere solo due dimensioni.

Ciò ha permesso di calcolare il numero totale di elementi dell'array originale, rimasto invariato, e la rispettiva dimensione, ossia 32.

Grazie a questa integrazione, si è stati in grado di ottenere valori plausibili di Chamfer Distance per ciascun modello da verificare. Questi valori sono stati salvati accuratamente all'interno di un foglio di calcolo, per la successiva fase di validazione e analisi tra i risultati ottenuti durante questa ricerca e quelli presentati nello studio precedente.

6 Test e Risultati ottenuti

6.1 Selezione dei modelli

Generalmente, dalla letteratura emergono alcuni modelli di oggetti che ricorrono frequentemente nei dataset di immagini sintetiche utilizzati nella ricerca scientifica. Questi modelli, come sedie, tavoli, armadi, librerie, auto, sono considerati categorie “standard”. Queste categorie rappresentano una vasta gamma di oggetti comuni, come sedie, aerei e automobili, e possono essere considerati dei benchmark affidabili per valutare le prestazioni delle reti nella ricostruzione dei modelli 3D. Il concetto di utilizzare categorie standardizzate di oggetti nei dataset di immagini sintetiche è una pratica comune nella ricerca in computer vision e ricostruzione 3D, in quanto permette di valutare l’efficacia dei nuovi approcci rispetto a quelli esistenti e confrontare i diversi risultati. Per tale motivo, si è deciso di concentrarsi su queste categorie standard, evitando oggetti specifici. Tra i vari dataset utilizzati, quello più completo e ricco è ShapeNet [7], avente circa 270 categorie di oggetti.

L’obiettivo principale di questa ricerca è stato quello di valutare le prestazioni e la qualità delle ricostruzioni tra il modello Pix2Vox e il modello BSP-Net, valutato nella tesi precedente. Una differenza fondamentale tra i due modelli era che mentre la rete BSP-Net aveva la capacità di generare direttamente una mesh 3D di un oggetto, la rete Pix2Vox produceva come output solo modelli 3D in formato voxelizzato. Nonostante ciò, per poter effettuare un confronto significativo tra i due modelli, è stato necessario utilizzare lo stesso dataset adottato per l’addestramento dell’autoencoder della rete BSP-Net.

Avendo deciso di utilizzare sia immagini reali di oggetti come input che il dataset Pix3D [35] per avere i corrispondenti modelli 3D e utilizzarli come “ground truth” nella fase di ricostruzione, per gli esperimenti condotti con BSP-Net si sono selezionate soltanto alcune categorie di oggetti: sedie, tavoli e librerie/armadi. Mentre, per questa ricerca il modello Pix2Vox è stato addestrato esclusivamente sulla categoria specifica delle sedie.

6.2 Esperimenti

6.2.1 Dimensione delle immagini

Gli esperimenti iniziali si sono concentrati sulla risoluzione delle immagini utilizzate per addestrare la rete neurale. Per l'analisi sono state selezionate due dimensioni specifiche: 128×128 px e 224×224 px (figura 6.1).



Figura 6.1: Rappresentazione di un'immagine sintetica di una sedia a risoluzione 128×128 px e 224×224 px.

Da diversi studi precedenti riguardanti reti neurali addestrate su dataset di immagini sintetiche, è emerso che molte reti neurali utilizzano immagini di dimensione 128×128 px per l'addestramento, di conseguenza tale risoluzione è stata adottata come punto di partenza per le indagini condotte in questa ricerca. Partendo da una bassa risoluzione di 128×128 px, si è deciso di condurre ulteriori esperimenti anche con risoluzioni più elevate per capire se immagini di dimensioni maggiori avrebbero potuto influire maggiormente nel processo di ricostruzione 3D.

In particolare, la risoluzione 224×224 px è stata presa in considerazione poiché è la risoluzione a cui opera nativamente la rete Pix2Vox. Pertanto, è stato ritenuto utile condurre valutazioni anche su questa particolare risoluzione delle immagini per comprendere appieno il suo impatto sul processo di ricostruzione e verificare se portasse a risultati significativamente migliori rispetto alla risoluzione più bassa.

I quattro dataset di immagini sintetiche, realizzati precedentemente e applicati ad una rete neurale con una struttura diversa da ResNet-18, mantengono fissi i parametri come materiali, illuminazione e posizione della camera, concentrandosi esclusivamente sulla valutazione dell'effetto della risoluzione delle immagini.

6.2.2 Distanza della camera

Un altro aspetto considerato riguarda la posizione della camera, nello specifico, ci si è concentrati sulla sua distanza, fissa o variabile, dall'oggetto 3D.

I dataset utilizzati non solo contengono immagini in cui la camera si muove attorno all'oggetto mantenendo costante la sua distanza, ma anche immagini in cui la camera variava in modo casuale la sua distanza dall'oggetto (figura 6.2). I diversi esperimenti sono stati condotti per valutare l'effetto della variazione della distanza della camera, consentendo così una comprensione più approfondita di come la prospettiva influenzi la rappresentazione tridimensionale degli oggetti.



Figura 6.2: Rappresentazione di un modello con distanza fissa (a sinistra) e variazione della camera (a destra).

6.2.3 Materiali e texture

Per esaminare l'impatto dei vari materiali e delle texture applicate ai modelli 3D, sono stati condotti diversi esperimenti per esaminare quanto queste variabili potessero influire sulla qualità di ricostruzione. In particolare, si sono utilizzati due dataset per ogni categoria di oggetti: uno contenente immagini di oggetti

con materiali applicati e un altro con immagini di oggetti senza materiali (figura 6.3).



Figura 6.3: Immagine di uno stesso modello senza materiale (a sinistra) e con applicazione del materiale (a destra).

Dopo aver effettuato i primi esperimenti in maniera generale sull'applicazione dei diversi materiali, ognuno costituito da diverse "mappe" per generare specifici effetti durante la fase di rendering, si è passati ad un'analisi più specifica. Per i successivi test, sono stati utilizzati vari dataset, ognuno dei quali conteneva una singola mappa, come la normal map o la roughness map, oppure solo la semplice color map, rappresentante il colore del materiale. Questo approccio è stato adottato per comprendere quanto l'assenza o la presenza di queste texture potesse influenzare l'efficacia della rete neurale. Inoltre, ha permesso anche di valutare l'impatto specifico di ciascuna mappa sul funzionamento del modello, fornendo così una visione più dettagliata sull'importanza delle texture.

Un'ulteriore analisi è stata fatta per individuare se ci fossero materiali particolarmente efficaci rispetto ad altri per i modelli 3D. Poter esaminare la presenza di texture con specifici pattern (vedi figura 6.4) ha permesso di capire se quest'ultimi potessero avere un impatto negativo, soprattutto durante la fase di addestramento della rete neurale. Questa analisi ha permesso di valutare l'influenza di determinati pattern di texture sui risultati del processo di addestramento. Tale indagine ha contribuito a identificare quali pattern e texture fossero più adatte a migliorare l'efficacia dell'addestramento della rete.



Figura 6.4: Esempio di un modello con pattern.

6.2.4 Illuminazione

Un altro parametro analizzato è stato l'illuminazione degli oggetti. In particolare, sono stati definiti due modelli di illuminazione. Il primo modello era costituito da luci posizionate in modo da illuminare uniformemente l'intero modello 3D, mentre nel secondo modello, le luci seguivano la disposizione del three-point lighting, con due luci di fronte al modello 3D e una dietro di esso, simulando un effetto più realistico grazie alla presenza di ombre.

Un'ulteriore analisi è stata condotta per valutare l'impatto della variazione dell'intensità delle luci sul risultato finale. Oltre ad utilizzare i primi due modelli, in cui le luci avevano un'intensità costante, si è utilizzato anche un dataset in cui l'intensità della luce variava per ogni singola immagine, generandone alcune più scure e altre più luminose. Anche in questo caso, le varie analisi hanno permesso di comprendere come la variazione dell'illuminazione potesse influenzare il processo di addestramento e di ricostruzione della rete neurale.

6.3 Risultati ottenuti

Gli esperimenti effettuati per questa ricerca hanno riguardato una specifica categoria di oggetti: le sedie.

Di seguito vengono presentati i risultati ottenuti e un confronto tra quest'ultimi e quelli ottenuti nella tesi precedente per valutare l'andamento della Chamfer Distance al variare dei diversi parametri.

6.3.1 Dimensione dell'immagine

Nel corso dei primi esperimenti condotti è stata valutata l'influenza delle dimensioni delle immagini utilizzate per addestrare la rete neurale. Per quest'ultimi sono stati utilizzati tre diversi dataset di immagini:

1. Il primo dataset conteneva immagini sintetiche create mantenendo la camera a una distanza fissa dal modello 3D.
2. Il secondo dataset comprendeva immagini generate applicando agli stessi modelli 3D i materiali utilizzati nel primo dataset, ma variando la distanza della camera dal modello.
3. Il terzo dataset conteneva immagini create utilizzando le posizioni della camera del secondo dataset, ma rimuovendo tutti i materiali dai modelli 3D.

Un esempio di immagini presenti nei tre dataset utilizzati è mostrato in figura 6.5.

Ogni dataset è stato generato in diverse risoluzioni, nello specifico con immagini a dimensioni 128×128 px e 224×224 px.

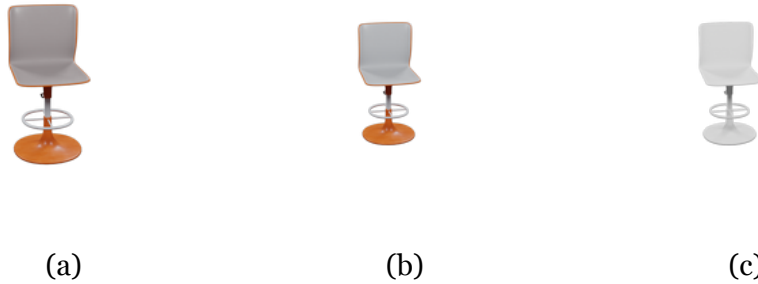


Figura 6.5: Immagini presenti nel (a) primo dataset, (b) secondo dataset e (c) terzo dataset.

I risultati ottenuti dai tre diversi dataset per la rete neurale Pix2Vox e il relativo confronto con la rete BSP-Net sono riportati nelle tabelle seguenti per le diverse risoluzioni delle immagini. Ogni tabella mostra le prestazioni delle due reti neurali per la categoria di oggetti analizzata.

In particolare, i risultati ottenuti per il primo dataset sono riportati in tabella 6.1, quelli relativi al secondo dataset in tabella 6.2 e quelli relativi al terzo dataset in tabella 6.3.

Chamfer Distance		
	Pix2Vox	BSP-Net
	224x224px	224x224px
Sedie	0,085	0,0110

Tabella 6.1: Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il primo dataset di immagini.

Chamfer Distance		
	Pix2Vox	BSP-Net
	224x224px	224x224px
Sedie	0,011	0,0047

Tabella 6.2: Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il secondo dataset di immagini.

Chamfer Distance				
Pix2Vox			BSP-Net	
	128x128px	224x224px	128x128px	224x224px
Sedie	0,0102	0,0081	0,0061	0,0055

Tabella 6.3: Confronto dei valori (arrotondati) di Chamfer Distance delle due reti neurali per il terzo dataset di immagini.

Esaminando attentamente i risultati per le diverse dimensioni, è emerso che all'aumentare della risoluzione, la Chamfer Distance diminuisce, indicando di conseguenza un miglioramento nella qualità della ricostruzione 3D.

Pix2Vox e BSP-Net sono entrambe reti neurali progettate per lavorare con immagini di dimensione 224x224px, ma differiscono notevolmente nell'architettura e nell'approccio alla generazione di modelli 3D. Pix2Vox si basa sul principio di generazione di volumi 3D da immagini 2D, attraverso l'uso di un'architettura che apprende le relazioni spaziali tra le diverse parti dell'oggetto nelle immagini e le utilizza per generare un modello 3D coerente. Al contrario della rete BSP-Net, basata sul modello di ResNet-18, che riduce progressivamente le dimensioni delle feature map attraverso strati di convoluzione e pooling, il modello Pix2Vox non esegue un pooling finale ma mantiene la dimensione delle feature map fino alla fase di decodifica, dove le informazioni spaziali vengono utilizzate per generare il modello 3D finale. L'aumento delle dimensioni delle immagini di input in Pix2Vox porterebbe a feature map più grandi nelle fasi intermedi dell'architettura, che vengono poi utilizzate per generare modelli 3D più dettagliati e coerenti.

In generale, si è osservato che l'aumento delle dimensioni delle immagini in un dataset, inizialmente realizzato per allenare una rete neurale basata sul modello di ResNet-18, porta a dei miglioramenti nei risultati anche quando tali immagini vengono utilizzate per una rete con un'architettura differente. Di conseguenza, una maggiore risoluzione delle immagini può avere un impatto positivo sulle prestazioni del modello, indipendentemente dalla specifica architettura utilizzata.

Per quanto riguarda i tempi di addestramento, questi sono notevolmente influenzati dalla quantità di dati e dalle risorse hardware della macchina utilizzata. Il computer impiegato per l'implementazione della rete neurale era costituito da una scheda grafica Nvidia GeForce GTX 1060 con 6 GB di memoria dedicata, la

quale è stata utilizzata per i calcoli eseguiti dalla rete neurale durante l'addestramento.

Nello specifico, per immagini di dimensioni 128x128px, il tempo per ogni singola epoca è stato di circa 6 minuti, con un tempo totale per completare una sessione di training di circa 50 minuti. Aumentando le dimensioni delle immagini a 224x224px, il tempo per ogni epoca è cresciuto a circa 18 minuti, con una durata totale di circa 3 ore. È interessante notare che questi tempi sono significativamente superiori rispetto alla rete BSP-Net. Infatti, utilizzando immagini con le stesse dimensioni (128x128px o 224x224px) la sessione di training con BSP-Net richiedeva solo 25/30 minuti, molto meno rispetto alla rete attuale.

I risultati di questo confronto sono riportati nella tabella 6.4.

Tempi di training medi				
	Pix2Vox		BSP-Net	
	128x128px	224x224px	128x128px	224x224px
Tempo per singola epoch	6 (m)	18 (m)	5,4 (s)	5 (s)
Tempo totale di training	3 (h)	3 (h)	24,27 (m)	29,17 (m)

Tabella 6.4: Confronto dei tempi medi di training delle due reti neurali per i tre dataset di immagini.

6.3.2 Distanza della camera e presenza di materiali

Un ulteriore risultato significativo riguarda la posizione della camera rispetto all'oggetto 3D. Per quanto riguarda Pix2Vox, il miglioramento del 70% nella ricostruzione delle sedie quando la distanza della camera dall'oggetto varia indica una notevole capacità di adattamento del modello (tabella 6.5). D'altra parte, anche BSP-Net mostra un miglioramento del circa 50% al variare della posizione della camera (tabella 6.6).

	Distanza fissa	Distanza variabile
Pix2Vox		
	224x224px	224x224px
Sedie	0,085	0,011

Tabella 6.5: Valori di Chamfer Distance per Pix2Vox relativi a un dataset di immagini in cui la camera è mantenuta a una distanza fissa dal modello 3D e un dataset in cui la camera varia la sua distanza dal modello.

	Distanza fissa	Distanza variabile
Chamfer Distance		
BSP-Net		
	224x224px	224x224px
Sedie	0,0110	0,0047

Tabella 6.6: Valori di Chamfer Distance per BSP-Net relativi a un dataset di immagini in cui la camera è mantenuta a una distanza fissa dal modello 3D e un dataset in cui la camera varia la sua distanza dal modello.

Questo miglioramento potrebbe essere attribuito alla diversità dei dati di allenamento quando la posizione della camera cambia. Inoltre, un ulteriore contributo è dato dall'utilizzo di immagini reali per la ricostruzione. Le immagini reali utilizzate, di fatto, raramente presentano un oggetto alla stessa distanza o centrato perfettamente nell'immagine, e ciò permette ai modelli di generalizzare meglio le variazioni di prospettiva (figura 6.6). Tali variazioni durante l'addestramento permettono di apprendere rappresentazioni più complesse, mitigando così il rischio di overfitting.



Figura 6.6: Esempio di immagini dal dataset di Pix3D impiegate nel processo di ricostruzione degli oggetti.

In generale, questi risultati suggeriscono che entrambe le reti neurali hanno dimostrato di essere altamente adattabili alle variazioni prospettiche dovute ai cambiamenti nella posizione della camera, con Pix2Vox che mostra un particolare miglioramento significativo nell'accuratezza delle ricostruzioni.

In merito alla presenza o meno di un materiale applicato all'oggetto, si è notata una notevole differenza nei risultati delle due reti neurali. Per Pix2Vox, la presenza di materiale sembra influire negativamente sulla precisione della ricostruzione, infatti nel caso di immagini rappresentanti oggetti senza materiale la Chamfer Distance è significativamente più bassa (indicando una migliore ricostruzione) rispetto alle immagini raffiguranti oggetti con materiale applicato (tabella 6.7). Al contrario, per BSP-Net l'aggiunta di materiale sembra migliorare leggermente la precisione nella ricostruzione (tabella 6.8).

Questi risultati suggeriscono che l'impatto del materiale sulla precisione della ricostruzione varia a seconda dell'architettura della rete neurale, delle specifiche caratteristiche e capacità nel gestire informazioni relative ai materiali.

	Modelli con materiale	Modelli senza materiale
Chamfer Distance		
Pix2Vox		
	224x224px	224x224px
Sedie	0.0112	0.0081

Tabella 6.7: Valori di Chamfer Distance per Pix2Vox relativi a un dataset di immagini ottenuti da modelli con materiale applicato e un dataset in cui i modelli sono senza materiale.

	Modelli con materiale	Modelli senza materiale
Chamfer Distance		
BSP-Net		
	224x224px	224x224px
Sedie	0,0047	0,0055

Tabella 6.8: Valori di Chamfer Distance per BSP-Net relativi a un dataset di immagini da modelli con materiale applicato e un dataset in cui i modelli sono senza materiale.

6.3.3 Materiali e texture

Ulteriori esperimenti sono stati condotti concentrandosi specificamente sulle caratteristiche dei materiali e delle texture applicate agli oggetti, al fine di comprendere come queste potessero influire sul corretto funzionamento della rete neurale.

Nello specifico, sono stati utilizzati diversi dataset di immagini sintetiche costituiti dall'aggiunta o rimozione di specifiche texture dai materiali, in modo tale da valutare come la presenza o l'assenza di ogni specifica "mappa" potesse influenzare il processo di addestramento e la ricostruzione degli oggetti 3D.

Nel primo dataset le immagini dei modelli contenevano solo il colore base del materiale, o color map, senza l'aggiunta di altre mappe. Le tabelle mostrano i risultati per entrambi le reti, rispettivamente per Pix2Vox (tabella 6.9) e BSP-Net (tabella 6.10).

Materiale realistico	Base color
Chamfer Distance	
Pix2Vox	
224x224px	224x224px
Sedie	0,0112
	0,0102

Tabella 6.9: Valori di Chamfer Distance per Pix2Vox relativo ad immagini con materiale “realistico” e immagini con solo la color map.

Materiale realistico	Base Color
Chamfer Distance	
BSP-Net	
224x224px	224x224px
Sedie	0,0047
	0,0046

Tabella 6.10: Valori di Chamfer Distance per BSP-Net relativo ad immagini con materiale “realistico” e immagini con solo la color map.

Osservando e confrontando i risultati ottenuti per le due reti neurali, si può evidenziare come per entrambe risulta esserci un miglioramento quando si utilizzano solo texture di base, come la color map, anziché texture più complesse. Tale miglioramento potrebbe essere attribuito alla mancanza di riflessioni luminose dovute a determinati valori del parametro “specular” nei materiali. Pertanto, in determinate situazioni, ridurre la complessità delle texture e limitare le riflessioni luminose può portare a una leggera miglioramento nella precisione della ricostruzione.

Successivamente, sono state aggiunte progressivamente le altre “mappe”, come la normal map e la roughness map. In questi casi, l’inclusione di mappe aggiuntive, ha portato a risultati sostanzialmente simili a quelli ottenuti utilizzando solo la color map. Questo evidenzia che, per la rete Pix2Vox l’aggiunta di mappe ha portato ad un ulteriore miglioramento delle prestazioni, riducendo la Chamfer Distance. Questo potrebbe essere dovuto al fatto che queste mappe forniscono informazioni dettagliate sulla forma e sulla superficie degli oggetti, consentendo alla rete di apprendere caratteristiche più precise durante il training. Mentre,

per la rete BSP-Net l'aggiunta di dettagli come le informazioni sulla normale delle superfici o sulla rugosità non ha comportato miglioramenti significativi nella qualità della ricostruzione rispetto all'utilizzo solo della color map. Le tabelle di seguito mostrano i risultati per entrambe le reti, rispettivamente per Pix2Vox (Tabella 6.11) e BSP-Net (Tabella 6.12).

Base Color e Normal map		Base Color e Normal e Rough map	
Chamfer Distance			
Pix2Vox			
224x224px		224x224px	
Sedie	0,0087		0,0087

Tabella 6.11: Valori di Chamfer Distance per Pix2Vox relativi al dataset di immagini aggiungendo le altre mappe ai materiali.

Base Color e Normal map		Base Color e Normal e Rough map	
Chamfer Distance			
BSP-Net			
224x224px		224x224px	
Sedie	0,0047		0,0047

Tabella 6.12: Valori di Chamfer Distance per BSP-Net relativi al dataset di immagini aggiungendo le altre mappe ai materiali.

Un'analisi ulteriore è stata condotta utilizzando materiali che presentassero texture con specifici pattern (tabella 6.13). Sia per la rete Pix2Vox che BSP-Net, è stato osservato che i risultati peggiorano quando si utilizzano materiali con pattern molto evidenti.

Pattern		
Chamfer Distance		
	Pix2Vox	BSP-Net
	224x224px	224x224px
Sedie	0,0102	0,0048

Tabella 6.13: Confronto dei valori di Chamfer Distance per le due reti relativi al dataset di immagini con specifici pattern.

6.3.4 Illuminazione

Nei test successivi, sono state esaminate due diverse configurazioni di illuminazione, illuminazione omogenea e un modello di illuminazione “tree-point lighting”, applicate alla scena e ai modelli 3D utilizzati per generare le immagini sintetiche, nello specifico l’illuminazione omogenea e un’altra basata sul modello. I risultati per la rete neurale Pix2Vox sono mostrati nella tabella 6.14, mentre per la rete BSP-net nella tabella 6.15.

	Illuminazione omogenea	Three-point Lighting
Chamfer Distance		
Pix2Vox		
	224x224px	224x224px
Sedie	0,0132	0,0097

Tabella 6.14: Valori di Chamfer Distance per Pix2Vox relativi ad esperimenti con modelli illuminati in maniera omogenea e modelli illuminati con la configurazione del three-point lighting.

	illuminazione omogenea	Three-point Lighting
	Chamfer Distance	
	BSP-Net	
	224x224px	224x224px
Sedie	0,0047	0,0048

Tabella 6.15: Valori di Chamfer Distance per BSP-Net relativi ad esperimenti con modelli illuminati in maniera omogenea e modelli illuminati con la configurazione del three-point lighting.

Per entrambi le reti, è emerso che un’illuminazione omogenea del modello 3D tende a migliorare le prestazioni durante la fase di ricostruzione, a differenza di un’illuminazione più realistica. Questa differenza potrebbe essere attribuita al fatto che un’illuminazione omogenea previene la formazione di aree “più scure” o ombre nette sugli oggetti, le quali possono essere interpretate dalla rete come possibili caratteristiche durante il processo di addestramento.

Inoltre, è interessante notare che la rete BSP-Net sembra essere meno influenzata dall’illuminazione omogenea rispetto alla rete Pix2Vox. Questo potrebbe essere dovuto alla capacità della rete BSP-Net di gestire in modo più efficace le informazioni di illuminazione, grazie ad una migliore capacità di adattamento alle variazioni nell’illuminazione del modello.

Tuttavia, sono ancora in corso i test per le immagini con dimensioni 128x128px.

6.4 Confronto grafico

I risultati sperimentali derivanti da diverse prove condotti attraverso l’addestramento della rete Pix2Vox (figura 6.7) e i risultati ottenuti in precedenza attraverso la ricerca sulla rete BSP-Net (figura 6.8), sono accuratamente rappresentati nei due diversi grafici sottostanti.

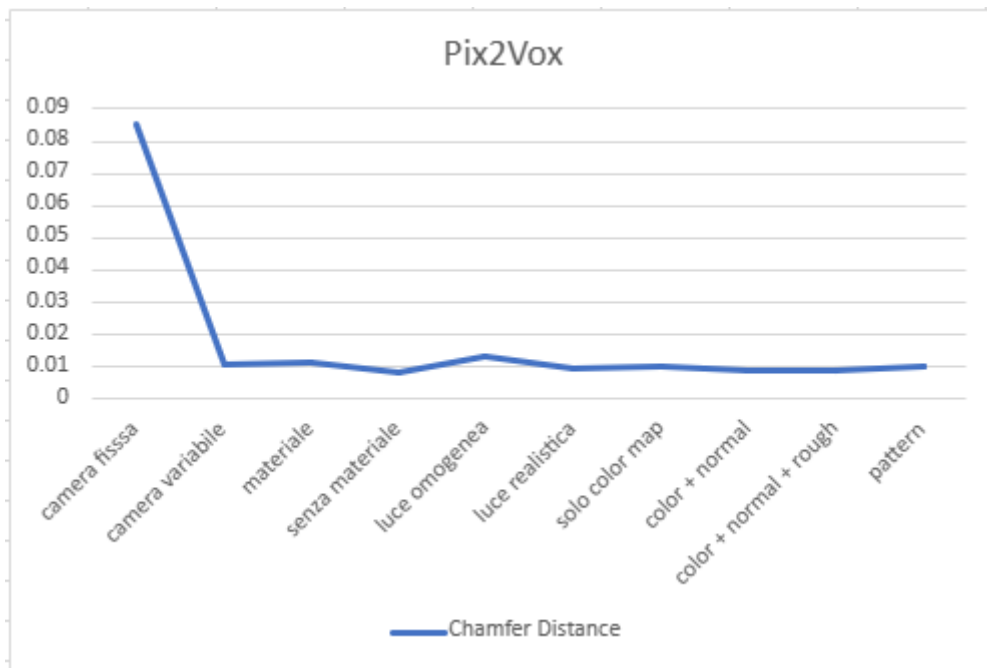


Figura 6.7: Analisi dell'andamento della Chamfer Distance, per la rete Pix2Vox, al variare dei parametri chiave.

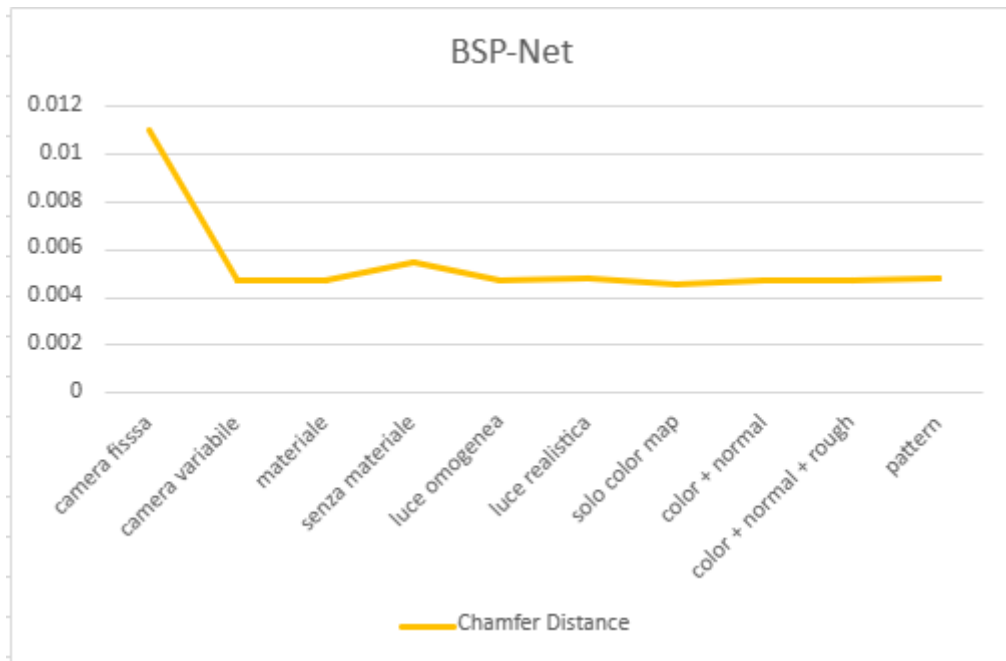


Figura 6.8: Analisi dell'andamento della Chamfer Distance, per la rete BSP-Net, al variare dei parametri chiave.

Andando ad analizzare da più vicino l'andamento di tale metrica al variare dei vari parametri e il relativo confronto tra le due reti, rispettivamente Pix2Vox e BSP-Net si possono notare alcune tendenze interessanti. Per quanto riguarda la posizione della camera, nel caso di immagini in cui la camera è fissa Pix2Vox ha difficoltà a gestire la profondità; al contrario, quando la posizione della fotocamera è variabile, Pix2Vox è più robusto nel gestire diverse angolazioni di visualizzazione.

Per quanto riguarda materiali e texture, la presenza di materiali sembra influenzare entrambe le reti in modo simile, al contrario Pix2Vox mostra una performance leggermente migliore rispetto a BSP-Net quando non sono presenti materiali.

In particolare, l'aggiunta di mappe come la normal map e la rough map ha portato a un ulteriore miglioramento delle prestazioni della rete Pix2Vox. Al contrario, per la rete BSP-Net l'aggiunta di queste mappe ha un impatto limitato, potendo già essere in grado di catturare abbastanza dettagli dalle mappe di base.

Nel caso in cui vengono utilizzate solo le color map o specifici pattern, entrambe

le reti mostrano risultati simili.

Inoltre, in presenza di luce realistica, le reti hanno performance simili con valori di Chamfer Distance relativamente bassi.

Queste variazioni indicano che le due reti rispondono in modo diverso alle condizioni specifiche del problema e suggeriscono che l'efficacia di una rete può variare notevolmente in base al contesto specifico in cui viene utilizzata.

7 Conclusioni e sviluppi futuri

7.1 Conclusioni

L'obiettivo principale di questo studio è stato condurre una serie di esperimenti al fine di effettuare un'analisi comparativa sulla precisione della ricostruzione di modelli 3D attraverso l'utilizzo della metrica Chamfer Distance. In particolare, è stato esteso ed applicato l'approccio metodologico delineato nella tesi precedente ad un modello architettonico diverso da ResNet-18. Tuttavia, l'intenzione è stata quella di analizzare l'andamento di questa metrica al variare dei parametri fondamentali che influenzano il processo di addestramento e, di conseguenza, la qualità della ricostruzione 3D, anziché confrontare i valori assoluti della Chamfer Distance tra gli esperimenti condotti.

Questo approccio ha permesso di confrontare l'efficacia e la robustezza di due reti neurali considerate in diverse condizioni, riuscendo a comprendere come variabili come la risoluzione delle immagini, la posizione della camera, la presenza di materiali e texture, l'illuminazione, influiscano sul processo di addestramento.

Nell'analizzare come questi parametri hanno un impatto significativo sulla capacità delle reti neurali di apprendere dai dati si sono osservate variazioni nei valori medi della Chamfer Distance, calcolata tra i modelli ground truth e quelli ricostruiti. In particolare, si è notato che all'aumentare della risoluzione dell'immagine, i tempi di training aumentano notevolmente, potendo affermare così che la risoluzione dell'immagine è un parametro critico che influenza sia la qualità della ricostruzione che il tempo necessario per addestrare il modello. Per quanto riguarda gli esperimenti relativi all'applicazione di diverse "mappe" ai materiali, ci sono state alcune variazioni nella Chamfer Distance, rendendo difficile determinare se queste ultime indicassero miglioramenti o peggioramenti significativi.

Tuttavia, esaminando dettagliatamente i risultati, è risultato evidente che Pix2Vox mostra una maggiore variabilità nelle sue prestazioni in diverse condizioni rispetto a BSP-Net. Ad esempio, in alcune situazioni specifiche, come quando la posizione della camera è variabile o in assenza di materiali, Pix2Vox sembra avere prestazioni migliori rispetto a BSP-Net. Al contrario, BSP-Net dimostra una maggiore coerenza e offre prestazioni superiori in condizioni generali, come quando la camera è in posizione fissa. Queste variazioni indicano che l'efficacia di una rete neurale può variare notevolmente in base al contesto specifico in cui viene utilizzata.

Le diverse risposte delle reti Pix2Vox e BSP-Net ai diversi parametri possono

essere dovute a vari fattori, tra cui:

- Architettura della rete. Una diversa architettura, con strati e meccanismi di apprendimento differenti, porta ad una diversa capacità di apprendere e interpretare le informazioni dai dati in input.
- Complessità del modello. Gli algoritmi utilizzati durante il training delle reti possono condizionare l'adattamento dei pesi utilizzati per minimizzare la funzione di perdita.
- Algoritmi di ottimizzazione. Gli algoritmi utilizzati durante il training delle reti possono condizionare l'adattamento dei pesi utilizzati per minimizzare la funzione di perdita.
- Iperparametri e regolazione. I valori degli iperparametri, come il tasso di apprendimento, possono influenzare la velocità e la stabilità durante il training.
- Sensibilità al contesto. Alcune reti possono essere più sensibili a specifiche caratteristiche dei dati rispetto ad altre, ad esempio una rete potrebbe essere più sensibile alla presenza o all'assenza di riflessioni luminose o texture complesse influenzando le sue prestazioni.
- Casualità. Durante il training di reti neurali, le inizializzazioni casuali dei pesi e l'ordine dei dati di addestramento possono introdurre variabilità nel processo di apprendimento, portando a risultati leggermente diversi in esecuzione rispetto al training.

Altri fattori che possono influenzare il processo di training di una rete neurale includono il tipo di hardware utilizzato, come CPU o GPU, le diverse versioni delle librerie di machine learning e i relativi parametri di configurazione, così come la qualità e l'accuratezza del preprocessing dei dati. Ognuno di questi elementi può avere un impatto significativo sulle prestazioni della rete durante il training e, di conseguenza, sulla qualità dei risultati ottenuti.

Inoltre, l'applicazione della pipeline proposta nella tesi precedente a una rete neurale con un'architettura diversa da ResNet-18 è risultata essere valida. Analizzando la Chamfer Distance, per questo caso studio, in diverse condizioni e con vari parametri, si è osservato che, a parte alcune minime variazioni, l'andamento risulta essere coerente con i risultati ottenuti in precedenza. Questa coerenza sottolinea la robustezza e l'efficacia della metodologia proposta, offrendo una base affidabile per ulteriori esplorazioni e applicazioni nel campo della ricostruzione 3D.

7.2 Eventuali miglioramenti e sviluppi futuri

Lo studio specifico condotto in questa tesi rappresenta un punto di partenza fondamentale per ulteriori revisioni e ricerche in questo campo. Sarebbe opportuno condurre ulteriori analisi al fine di validare in modo rigoroso i risultati ottenuti. Un approccio potenziale potrebbe prevedere l'addestramento della rete utilizzando categorie di oggetti diverse da quelle considerate in questo studio. Inoltre, sarebbe opportuno valutare l'estensione della rete Pix2Vox per la ricostruzione di oggetti 3D a partire da immagini RGB-D. Questa metodologia più ampia e diversificata potrebbe contribuire a una comprensione più approfondita e completa dei concetti esplorati, offrendo spunti preziosi per future ricerche nel campo in questione.

Di seguito sono elencati eventuali miglioramenti e sviluppi che potranno essere di aiuto per ampliare o validare ulteriormente questa ricerca:

- Analisi più approfondita sulle immagini reali utilizzate per la ricostruzione, prese in maniera casuale da quelle offerte dal dataset di Pix3D. Tramite ciò si potrebbe esaminare in modo specifico come i parametri delle immagini reali, come le dimensioni e la necessità di ridimensionamento per adattarle alle richieste della rete, influenzino la qualità della ricostruzione degli oggetti 3D.
- Ricerca più approfondita sull'influenza dei diversi materiali e texture sulla performance della rete neurale potrebbe essere condotta, cercando di comprendere al meglio come la variazione nei materiali influenzi il processo di ricostruzione. Questa analisi potrebbe estendersi anche alla valutazione dell'effetto dei vari colori sul risultato della ricostruzione.
- Provare a variare gli iperparametri specifici della rete neurale, come la batch size e il learning rate, in base alle dimensioni delle immagini utilizzate per l'addestramento. Inoltre, confrontare la qualità delle ricostruzioni 3D considerando anche le diverse dimensioni di ritaglio delle immagini potrebbe fornire indicazioni preziose sulla configurazione ottimale della rete.
- Analisi sulla ricostruzione Multi-view, ovvero esaminare immagini che mostrano oggetti da diverse prospettive, potrebbe portare a una comprensione più approfondita delle sfide e delle opportunità nella ricostruzione 3D da diverse angolazioni visive.
- Analisi specifica sulle reti neurali che effettuano la ricostruzione 3D a partire da immagini RGB-D, prendendo in considerazione il parametro di pro-

fondità e come quest'ultima potrebbe influenzare sulla precisione e la completezza delle ricostruzioni.

- Continuare a esplorare architetture più recenti o personalizzate per vedere se portano a miglioramenti significativi nella ricostruzione 3D, applicando a quest'ultime lo stesso approccio metodologico utilizzato in questo studio per avere ulteriori conferme o analizzare eventuali cambiamenti.

Tuttavia, considerando l'importanza dei diversi parametri chiave nell'influenzare le reti neurali durante il processo di ricostruzione 3D, l'elaborazione di un approccio metodologico robusto e architettrualmente indipendente rappresenta una prospettiva interessante.

Un tale approccio potrebbe coinvolgere diversi aspetti. A partire dall'analisi dettagliata sui parametri fondamentali e identificando i più influenti sull'addestramento, si potrebbero creare delle metriche di valutazione unificate che tengano conto di quest'ultimi e siano in grado di misurare l'adattabilità di un modello di ricostruzione 3D a diverse condizioni di input. Iniziare ad esplorare tecniche di addestramento adattive che possano regolare automaticamente i pesi e gli iperparametri della rete in base alle caratteristiche dell'immagine di input, permettendo così alla rete di adattarsi dinamicamente alle variazioni nei parametri delle immagini.

L'implementazione di questo approccio potrebbe portare a progressi significativi nella realizzazione di reti neurali per la ricostruzione 3D robusti e flessibili, capaci di adattarsi in modo dinamico alle varie condizioni di input.

A Requisiti della workstation e versione dei software

A.1 Workstation

Le attività condotte in questa tesi sono state eseguite impiegando una workstation dotata delle seguenti specifiche tecniche:

- Sistema Operativo: Windows 10 Pro - 64 bit
- Processore (CPU): Intel Core i7-8700 3.20GHz
- Memoria RAM: 16GB
- Scheda Grafica (GPU): NVIDIA GeForce GTX 1060 con 6GB di memoria dedicata

A.2 Software

Dal punto di vista del software, sono stati utilizzati i seguenti programmi con le rispettive versioni:

- Blender - versione 2.83.13 LTS
- VSCode - versione 1.73.1
- Python - versione 3.6.8 con le seguenti librerie:
 - PyTorch - versione 1.6.0 + CUDA versione 10.1
 - Torchvision - versione 0.11.3
 - TensorboardX - versione 1.2
 - Numpy - versione 1.19.5
 - OpenCV - versione 4.7.0.72
 - Matplotlib - versione 3.3.4
 - Scipy - versione 1.5.4

Riferimenti bibliografici

- [1] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid icp algorithms for surface registration. 2007. URL <https://ieeexplore.ieee.org/document/4270190>.
- [2] Binvox. <https://www.patrickmin.com/binvox/>.
- [3] Blender. <https://www.blender.org/>.
- [4] Blenderkit. <https://www.blenderkit.com>.
- [5] CGTrader. <https://www.cgtrader.com>.
- [6] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3d generative adversarial networks. *CoRR*, abs/2112.07945, 2021. URL <https://arxiv.org/abs/2112.07945>.
- [7] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. URL <http://arxiv.org/abs/1512.03012>.
- [8] Z. Chen, A. Tagliasacchi, and H. Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *CoRR*, abs/1911.06971, 2019. URL <http://arxiv.org/abs/1911.06971>.
- [9] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. *CoRR*, abs/1604.00449, 2016. URL <http://arxiv.org/abs/1604.00449>.
- [10] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam. Blenderproc. *CoRR*, abs/1911.01911, 2019. URL <http://arxiv.org/abs/1911.01911>.
- [11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.
- [12] B. Huang, J. Zhao, and J. Liu. A survey of simultaneous localization and mapping. *CoRR*, abs/1909.05214, 2019. URL <http://arxiv.org/abs/1909.05214>.

- [13] M. Li and H. Zhang. D²im-net: Learning detail disentangled implicit fields from single images. *CoRR*, abs/2012.06650, 2020. URL <https://arxiv.org/abs/2012.06650>.
- [14] Matplotlib. <https://matplotlib.org>.
- [15] Maya. <https://www.autodesk.it/products/maya>.
- [16] L. M. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. *CoRR*, abs/1812.03828, 2018. URL <http://arxiv.org/abs/1812.03828>.
- [17] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41, 1995. URL <https://api.semanticscholar.org/CorpusID:1671874>.
- [18] Numpy. <https://numpy.org>.
- [19] Open3D. <http://www.open3d.org>.
- [20] K. O’Shea and R. Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. URL <http://arxiv.org/abs/1511.08458>.
- [21] O. Özyesil, V. Voroninski, R. Basri, and A. Singer. A survey on structure from motion. *CoRR*, abs/1701.08493, 2017. URL <http://arxiv.org/abs/1701.08493>.
- [22] I. Piparo. Creazione di dataset di immagini sintetiche per l’addestramento di reti neurali dedicate alla generazione automatica di oggetti 3d, 2022. URL <http://webthesis.biblio.polito.it/id/eprint/25431>.
- [23] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. P. Eriksson, and C. Fookes. Image2mesh: A learning framework for single image 3d reconstruction. *CoRR*, abs/1711.10669, 2017. URL <http://arxiv.org/abs/1711.10669>.
- [24] S. Popov, P. Bauszat, and V. Ferrari. Corenet: Coherent 3d scene reconstruction from a single RGB image. *CoRR*, abs/2004.12989, 2020. URL <https://arxiv.org/abs/2004.12989>.
- [25] Python. <https://www.python.org>.
- [26] PyTorch. <https://pytorch.org>.
- [27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. URL <http://arxiv.org/abs/1612.00593>.

- [28] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Rezatofighi_Generalized_Intersection_Over_Union_A_Metric_and_a_Loss_for_CVPR_2019_paper.html.
- [29] O. Rukundo. Effects of image size on deep learning. *CoRR*, abs/2101.11508, 2021. URL <https://arxiv.org/abs/2101.11508>.
- [30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei. Image-net large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL <http://arxiv.org/abs/1409.0575>.
- [31] C. Sabottke and B. Spieler. The effect of image resolution on deep learning in radiography. *Radiology: Artificial Intelligence*, 2:e190015, 01 2020. doi: 10.1148/ryai.2019190015. URL https://www.researchgate.net/publication/338758547_The_Effect_of_Image_Resolution_on_Deep_Learning_in_Radiography.
- [32] A. Siddique and S. Lee. Sym3dnet: Symmetric 3d prior network for single-view 3d reconstruction. *Sensors*, 22(2), 2022. ISSN 1424-8220. URL <https://www.mdpi.com/1424-8220/22/2/518>.
- [33] Sketchfab. <https://sketchfab.com>.
- [34] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. *CoRR*, abs/1804.04610, 2018. URL <http://arxiv.org/abs/1804.04610>.
- [35] X. Sun, J. Wu, X. Zhang, Z. Zhang, C. Zhang, T. Xue, J. B. Tenenbaum, and W. T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. *CoRR*, abs/1804.04610, 2018. URL <http://arxiv.org/abs/1804.04610>.
- [36] TensorFlow. <https://www.tensorflow.org>.
- [37] Turbosquid. <https://www.turbosquid.com>.
- [38] Unity. <https://unity.com>.
- [39] M. J. van Kreveld, F. Staals, A. Vaxman, and J. L. Vermeulen. Approximating the earth mover’s distance between sets of geometric objects. *CoRR*, abs/2104.08136, 2021. URL <https://arxiv.org/abs/2104.08136>.

- [40] B. Vanherle, S. Moonen, F. V. Reeth, and N. Michiels. Analysis of training object detection models with synthetic data. 2022. URL <https://arxiv.org/abs/2211.16066>.
- [41] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *CoRR*, abs/2111.12702, 2021. URL <https://arxiv.org/abs/2111.12702>.
- [42] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. *IEEE Winter Conference on Applications of Computer Vision*, 2014. URL <https://doi.org/10.1109/wacv.2014.6836101>.
- [43] H. Xie, H. Yao, X. Sun, S. Zhou, S. Zhang, and X. Tong. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. *CoRR*, abs/1901.11153, 2019. URL <http://arxiv.org/abs/1901.11153>.
- [44] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann. DISN: deep implicit surface network for high-quality single-view 3d reconstruction. *CoRR*, abs/1905.10711, 2019. URL <http://arxiv.org/abs/1905.10711>.