Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Triennale in Informatica

GENERAZIONE AUTOMATICA DI PLANIMETRIE

Relatore:
Chiar.mo Prof.
PAOLO CIANCARINI

Presentata da:
LEON PIRAZZOLI

Sessione Straordinaria
Anno Accademico 2022-23

Sommario

Il mondo dell'architettura e quello dell'informatica si sono incrociati più volte nel corso degli anni dando vita a vere rivoluzioni tecnologiche il cui impatto può essere osservato nelle opere dei grandi architetti così come negli ambienti di tutti i giorni. Con l'avvento dell'era dell'intelligenza artificiale ci troviamo di fronte a una nuova potenziale rivoluzione del modo in cui vengono progettati e rappresentati gli ambienti in cui viviamo. In questa tesi esploreremo le tecniche innovative che appartengono a una fase fondamentale del processo architettonico: la generazione automatica di planimetrie. Esamineremo i due approcci principali basati sul deep learning descrivendone le tecniche, le strategie e i vantaggi. Nella seconda parte invece verrà presentato un prototipo che mette in pratica questi metodi per produrre automaticamente configurazioni di stanze partendo da un perimetro. Al termine di questa sperimentazione valuteremo l'efficacia degli approcci studiati ed esploreremo i loro casi d'uso, scoprendo che l'effetto di questa contaminazione tra intelligenza artificiale e architettura supera le barriere delle rispettive discipline e può offrire contributi in ambiti inusuali.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Nell'era digitale in continua evoluzione, l'applicazione dell'intelligenza artificiale rivoluziona molteplici settori, trasformando radicalmente le metodologie tradizionali. Uno dei campi che sta beneficiando in modo significativo di questa rivoluzione è l'architettura, dove l'introduzione di strumenti basati sull'IA sta apportando innovazioni sostanziali.

L'avvento dei primi strumenti di design elettronici negli anni 80, come i software di Computer Assisted Drawing (CAD), ha per sempre cambiato il mondo dell'architettura. I nuovi metodi proposti permisero agli architetti di lavorare a un livello più alto senza doversi più occupare di fasi del processo ripetitive e di basso livello. Con la nuova ascesa delle tecnologie di IA è possibile fare un altro passo nella stessa direzione astraendo ulteriormente il processo di design degli spazi.

Con planimetria si intende la rappresentazione bidimensionale dell'organizzazione interna di un edificio. Le informazioni contenute sono relative al perimetro dell'edificio, disposizione delle stanze, finestratura ed eventualmente arredamento di esempio. A livello grafico una planimetria è composta sia da disegni che da scritte e simboli che possono indicare l'orientamento della rappresentazione, scale, senso di apertura delle porte, materiali, etc...

L'automazione del processo di produzione di planimetrie per mezzo di computer prende il nome di Space Layout Planning (SLP) [13] o Automated

Space Layout Planning (ASLP) [12]. Questo settore è stato oggetto dei primi studi già negli anni 50 con Immer e Buffa (1), ma negli ultimi cinque anni ha visto una crescita considerevole grazie agli sviluppi nel campo dell'intelligenza artificiale

Questa tesi si propone di esplorare l'applicazione dell'intelligenza artificiale nella generazione automatica di planimetrie, un processo cruciale nella progettazione architettonica. Il lavoro sarà diviso in due parti: la prima sarà dedicata a una revisione sistematica dello stato dell'arte nel settore basata su metodo PRISMA, mentre la seconda sarà di natura sperimentale. Verrà presentata una *proof of concept* che applica le idee descritte nella prima parte e ne sonda le potenzialità di sviluppo.

Capitolo 2

I Metodi

Per gli scopi di questo studio sono stati analizzati dieci articoli sull'argomento. Di questi dieci, otto sono di carattere sperimentale e i restanti due sono revisioni sistematiche o indagini. Per la ricerca di queste fonti si è fatto uso di tre database di letteratura scientifica, rispettivamente Google Scholar, Web of Science e AlmaStart. Alcuni articoli sono stati esplorati in seguito alla loro citazione in altri lavori. Di seguito verranno presentati gli obbiettivi della ricerca e i metodi che hanno guidato la scelta delle fonti.

2.1 Obiettivi

La domanda di ricerca che guida questa indagine è:

Come si può utilizzare l'intelligenza artificiale per generare automaticamente una planimetria residenziale dato il perimetro dell'edificio?

Questa domanda può essere spezzata in diverse sotto-domande importanti per la scelta degli articoli da analizzare. Per prima cosa dobbiamo prendere in considerazione il fatto che l'intelligenza artificiale è un insieme di tecniche molto vasto e in continua espansione, per questo motivo è stato deciso di scegliere articoli che includono tecniche provenienti da rami diversi dell'intelligenza artificiale. In questo modo la panoramica sullo stato dell'ar-

te che presenteremo sarà più comprensiva possibile e può offrire soluzioni per versioni differenti dello stesso problema.

Un'altra domanda che occorre considerare è *che caratteristiche devono avere le planimetrie generate?* Se analizzassimo solo i lavori che producono planimetrie complete e arredate rischieremmo di lasciare indietro dei progetti che implementano tecniche meno esplorate ma altrettanto utili.

Per non restringere eccessivamente il campo di ricerca evitiamo di porre vincoli specifici su proprietà tecniche delle soluzioni come costo computazionale, dimensioni del dataset di allenamento e stabilità strutturale delle soluzioni. Si rivelerà quindi necessario valutare quali siano le opzioni più vantaggiose tra i diversi trade-off che si presentano. Nella prossima sezione verranno presentati i criteri di selezione applicati per filtrare le fonti utili a rispondere alla nostra domanda di ricerca.

2.2 Criteri di selezione

Per la selezione degli articoli presentati in questa indagine sono stati applicati filtri sulla data di produzione del documento e le parole chiave che lo rappresentano. Nel processo di scelta delle fonti si è cercato di creare una raccolta "orizzontale", cioè che include al suo interno una varietà di approcci di natura diversa.

Per trovare le soluzioni che compongono lo stato dell'arte attuale in questo settore sono stati escluse le fonti scritte al di fuori del range di date 2019 - 2023. La ricerca si è incentrata su documenti, scritti sia da architetti che da informatici, che contenessero le parole chiave "Artificial Intelligence", "floorplan", "spatial layout" e "generation". È stato necessario filtrare ulteriormente i risultati per escludere gli studi riguardanti la generazione di "floorplan" per circuiti elettronici. Per i nostri scopi è stato deciso di escludere studi riguardanti l'interpretazione o parsing di planimetrie; questo perché tecniche del genere sono utili in un sottoinsieme minore di approcci al problema e possono essere esplorate in studi futuri. Al contrario, sono stati

inclusi degli studi che includono tecniche di intelligenza artificiale per la rappresentazione di spazi ma che non hanno come obbiettivo principale la vera e propria generazione di planimetrie. Anche se in questi progetti la generazione può risultare come un effetto collaterale dei modelli utilizzati, le tecnologie presentate possono essere estratte e utilizzate per creare programmi che hanno come obbiettivo principale la generazione di nuove planimetrie. In alcuni casi questi approcci prendono in considerazione proprietà particolari del dataset utilizzato che non vengono trattate in altri studi sulla generazione, ma che possono offrire nuove regole su cui basare i modelli di generazione per conferire loro un nuovo modo di "vedere" e produrre le planimetrie. Questo è il caso di [2], che pur non avendo la produzione di nuove planimetrie come obbiettivo principale, usa tecniche che incorporano il comportamento umano nella rappresentazione delle planimetrie.

Nel prossimo capitolo classificheremo questi approcci che rappresentano l'attuale stato dell'arte e presenteremo degli esempi di applicazioni pratiche delle tecniche trattate.

Capitolo 3

Risultati

In questa sezione esploreremo le soluzioni proposte dai diversi studi considerati. Classificheremo gli approcci al problema in base alla tecnologia usata nella rappresentazione delle planimetrie per la loro elaborazione da parte di modelli di intelligenza artificiale. Divideremo quindi questo capitolo in due:

1. approccio basato sulle immagini
2. approccio basato sui grafi

L'approccio basato sulle immagini si basa sulla tecnologia di General Adversarial Network (GAN), mentre l'approccio basato sui grafi si basa su grafi e Graph Neural Network (GNN). Nonostante esistano altri metodi oltre a quelli sopra citati, è stata scelta questa coppia di approcci perché sono i più sviluppati e promettenti nella generazione di immagini. Altri gruppi di tecniche, come l'approccio basato sulla performance [12], sono più efficaci nell'analisi e classificazione di planimetrie.

Questo capitolo sarà strutturato come segue: avremo due sezioni dedicate agli approcci, ognuna di queste sarà divisa in una panoramica iniziale sui metodi, seguiranno alcune sotto-sezioni sugli strumenti e tecnologie utilizzate per la loro implementazione, poi due esempi di studi per ciascun approccio che applicano in pratica le tecniche viste e infine una conclusione che confronta queste strategie e ne analizza i vantaggi e casi d'uso.

3.1 Approccio basato sulle immagini

L'immagine è il mezzo più immediato e utilizzato per trasmettere informazioni sull'organizzazione di uno spazio, per questo motivo esiste una grossa quantità di immagini che può essere sfruttata come dataset di allenamento per un'intelligenza artificiale. Questa categoria si divide a sua volta in due gruppi di metodi: 1) approccio basato sull'apprendimento, 2) approccio basato sulle regole [2].

Il primo gruppo contiene alcuni tra i progetti più promettenti del settore, nelle prossime pagine ne analizzeremo due: il primo ([20]) segue un approccio naif, mentre il secondo ([5]) presenta una struttura più complessa. Entrambi utilizzano la stessa tecnologia, il software di mapping di immagini Pix2Pix ([10]): questo strumento possiede un'architettura basata su Generative Adversarial Network (GAN), un tipo di rete neurale capace di generare immagini molto vicine alla realtà.

3.1.1 GAN

Questa tecnologia appartiene al campo del Deep Learning (DL) ed è nata nel 2014 dal contributo di [8]. Le GAN sono una versione specializzata delle reti neurali (NN) in grado di generare dati idealmente indistinguibili da quelli usati per il suo addestramento.

La brillante idea dietro queste reti sta nella loro struttura. Una GAN ha due componenti: un generatore G e un discriminatore D. L'input di G è "rumore" casuale che viene trasformato in un oggetto in output cercando di minimizzare una loss function di partenza. Il compito di D sta nel ricevere due oggetti in input e distinguere quello reale. G e D sono collegati in modo che gli oggetti restituiti dal generatore vengano passati al discriminatore, che deve distinguerli da altri oggetti reali appartenenti al dataset di allenamento (3.1). A ogni iterazione il discriminatore aggiorna i parametri della loss function, in questo modo si instaura un feedback loop che porterà ad ottenere un output del generatore sempre più credibile e vicino al dataset di allenamento.

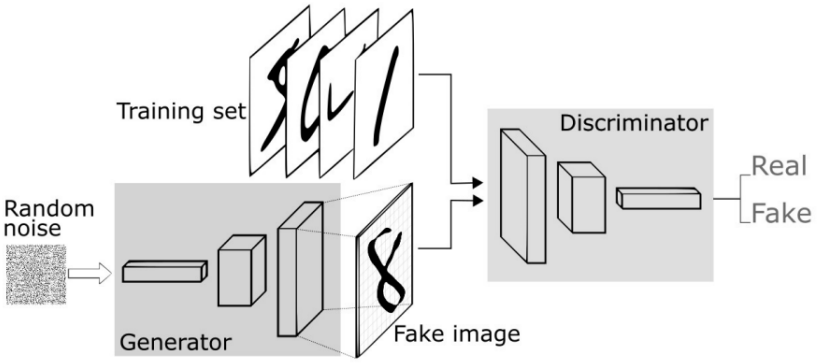


Figura 3.1: Architettura GAN [5]

3.1.2 Pix2Pix

Pix2Pix è un modello di intelligenza artificiale atto alla mappatura tra immagini. Per ogni immagine fornita, ne viene generata una in output secondo una qualche relazione con l'immagine in input. La relazione che definisce l'immagine in output è stabilita tramite l'addestramento del modello, durante il quale vengono fornite coppie di immagini con lo stesso tipo di relazione che vogliamo ottenere. Per esempio le coppie di immagini utilizzate per l'allenamento possono consistere in disegni di oggetti senza colori e gli stessi disegni colorati, in questo modo il modello produrrà immagini colorate dati i contorni del disegno.

Pix2Pix utilizza una versione specializzato di GAN chiamata Conditional GAN [10], in cui è possibile specificare una condizione che il generatore deve rispettare. La condizione scelta in questo caso è che l'output deve dipendere da un'immagine di input.

3.1.3 Approccio Naif

Nel 2020 Hao Zheng ha utilizzato Pix2Pix per creare due modelli generativi di planimetrie di appartamenti. I due modelli sono stati allenati su due dataset distinti: uno contiene 1279 immagini di piante di appartamenti

giapponesi, mentre il secondo contiene 112 planimetrie di appartamenti cinesi. I dataset sono composti dalle "coppie perimetro vuoto" - "planimetria annotata completa", mirando quindi alla generazione delle planimetrie in un solo step.

Il processo di allenamento è durato 100 epoche (un'epoca equivale a un ciclo completo attraverso l'intero dataset), al termine del quale sono state prodotte planimetrie con una divisione tra stanze riconoscibile. I modelli hanno quindi attraversato una fase di test, che consiste nella generazione di una planimetria partendo da un perimetro proveniente dal dataset di allenamento. I test effettuati hanno sì prodotto immagini in cui si riescono a distinguere stanze e mobili, ma in numerosi casi la planimetria generata non è strutturalmente solida a livello architettonico. In alcuni esempi presentati non è possibile discernere la funzione di alcune stanze a causa di artefatti e sfocature che minano la leggibilità dell'arredamento. In altri casi, soprattutto riscontrati nel modello allenato sulle planimetrie giapponesi, si può verificare l'inutile ripetizione di piccole stanze lungo i bordi dell'appartamento e la comparsa di caratteri illeggibili appartenenti alle annotazioni nelle planimetrie di allenamento, che rendono poco pratico l'uso da parte di architetti delle planimetrie generate. Gli autori concludono che per ottenere risultati migliori occorre utilizzare un dataset con uno stile di disegno uniforme e delle regole di design semplici.

Personalmente sono convinto che la qualità dei risultati sia imputabile al fatto che il modello debba generare una planimetria completa partendo dal perimetro in un solo step. Dividendo il processo in più fasi e semplificando i compiti di generazione si potrebbe ottenere una maggiore accuratezza. Per esempio anziché aspettarsi la generazione di disegni di mobili, si può considerare uno step intermedio con forme più semplici come macchie di colore, in modo da dare più importanza alla posizione degli oggetti. Analizzeremo ora un progetto che fa esattamente questo con risultati notevoli.

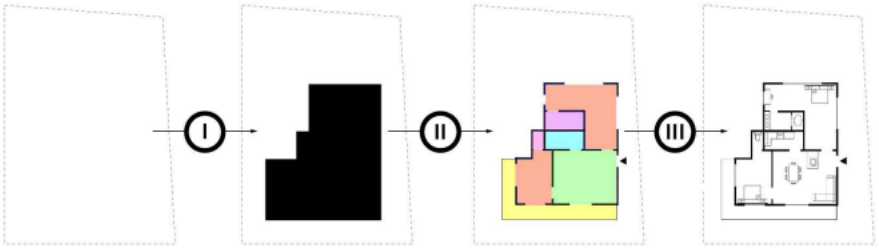


Figura 3.2: Pipeline di ArchiGAN [5]

3.1.4 ArchiGAN

Nel 2019 l'architetto francese Stanislas Chaillou ha creato ArchiGAN, un software in grado di prendere in input il perimetro di un edificio e restituire una planimetria arredata. Il programma è in realtà una pipeline composta da tre moduli che svolgono diverse fasi del processo. Il primo modulo si occupa di ricavare il perimetro di un appartamento partendo dall'area del lotto fornito dall'utente. Il secondo modulo usa l'output del primo per restituire una prima suddivisione in stanze, le stanze sono rappresentate da macchie di colore diverse in base alla loro funzione, in questo stage viene anche definita la finestrazione. A questo punto il terzo modulo si occupa dell'arredamento della stanza rappresentato da macchie di colore diverse per tipologia di mobili. Infine un ultimo sotto-modulo effettua il rendering della stanza passando da macchie di colore a disegni più definiti di mobili e muri.

Nel caso di ArchiGAN, il dataset utilizzato consiste in più di 700 planimetrie annotate di case e appartamenti della città di Boston. La pipeline di Chaillou è quindi composta da 3 istanze di Pix2Pix allenate sui diversi compiti di recupero del perimetro, divisione del perimetro in stanze e arredamento delle stanze (3.2). È da notare il fatto che questo approccio, nonostante l'efficacia, causa una triplicazione delle dimensioni del dataset. Infatti se un dataset normale consiste in un insieme di coppie di immagini di perimetri e planimetrie finite, in questo caso c'è bisogno di immagini iniziali dell'area di proprietà, immagini intermedie della suddivisione in stanze e immagini

delle stanze arredate dove i mobili sono rappresentati da macchie di colore. Questo fattore sarà da tenere a mente al momento della scelta di un approccio da seguire dato il suo impatto sulla grandezza del dataset e sui tempi di allenamento.

3.1.5 Stile

Un'altra potente tecnica che possiamo imparare da ArchiGAN, è la possibilità di generare planimetrie con uno stile preciso. Questo è possibile sfruttando l'emergere naturale del "bias" di un dataset di addestramento. Durante la fase di training del modello, le immagini usate hanno inevitabilmente alcune tendenze comuni che vengono catturate dal modello e riprodotte nelle immagini generate, questo comportamento prende il nome di "bias". Le immagini possono presentare tendenze indesiderate come la relegazione delle stanze di servizio (bagni, cucine, etc...) ai margini della planimetria, ma è lo stesso meccanismo che permette al modello di produrre piante strutturalmente solide. Questa capacità delle GAN di "intuire" le regole sottostanti alla costruzione di un'immagine corretta [12] è anche quello che ci permette di riprodurre un certo stile architettonico, dato che può essere ridotto a un insieme di regole e tendenze.

Per produrre un modello che replichi uno stile architettonico, occorre costruire un dataset di allenamento di planimetrie in quello stile. Per esempio ArchiGAN è in grado di replicare quattro stili: villa a schiera, Manhattan, barocco e vittoriano (3.3). Altri progetti hanno cercato di riprodurre lo stile di un singolo architetto come nel caso di GAN Hadid [18], il cui obbiettivo era di generare immagini di facciate di edifici nello stile della celebre architetta Zaha Hadid. I risultati sono poco soddisfacenti a causa, tra altre cose, dell'uso di un dataset a bassa risoluzione quindi povero di dettagli [4].

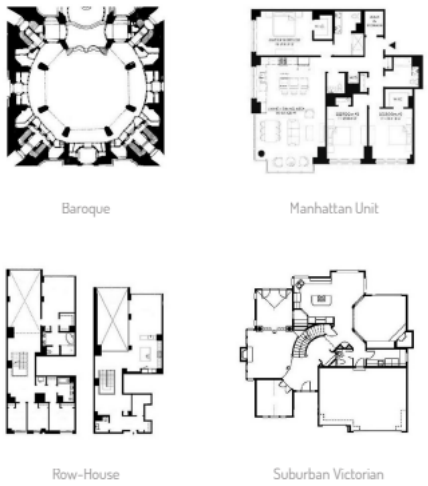


Figura 3.3: 4 stili diversi di ArchiGAN [5]

3.1.6 Conclusioni

L'approccio basato sulle immagini specializzato nell'apprendimento produce risultati impressionanti a livello di credibilità. Nonostante la potenziale accuratezza delle planimetrie generate ci sono diversi punti da analizzare in merito alla sua usabilità.

Per prima cosa la natura della tecnologia generativa utilizzata rende molto difficile (a volte impossibile) specificare dei requisiti aggiuntivi. Non possiamo indicare ai modelli caratteristiche come il numero di stanze o la presenza necessaria di elementi strutturali interni al perimetro come colonne o muri portanti. I progetti che utilizzano le GAN o Pix2Pix per generare una soluzione partendo dal perimetro in un solo step come Zheng et al. 2020 sono delle vere e proprie black box, cioè un prodotto su cui l'utente non ha alcun controllo dal momento in cui fornisce l'input al momento in cui è generato l'output. Questo annulla quasi completamente l'utilità del prodotto nel caso in cui l'architetto abbia dei requisiti specifici da rispettare. Nel caso di ArchiGAN l'architettura a pipeline rende il software una grey box ([5]), cioè l'utente ha la possibilità di intervenire in determinati punti del processo generativo. In ArchiGAN la separazione tra i diversi moduli permette l'inter-

vento dell'architetto nel caso in cui ci siano problemi tra una fase e l'altra, o se ci sono vincoli specifici da rispettare l'utente può produrre autonomamente una planimetria intermedia da sottoporre a selezionati passi del processo generativo.

Un'altra questione da considerare è la dimensione del dataset di allenamento e il potere computazionale richiesto per ottenere risultati soddisfacenti in tempi ragionevoli. Valutare la performance delle GAN è un compito difficile, infatti le GAN non hanno una loss function oggettiva, diversamente da altri modelli di DL che sono allenati usando una loss function fino alla convergenza. Non esiste quindi un modo di determinare oggettivamente il progresso dell'allenamento [20]. Questo significa che non possiamo dare una stima precisa del tempo impiegato per l'allenamento ma sappiamo che il tempo impiegato sarà direttamente proporzionale alle dimensioni del dataset [11]. In conclusione un approccio "grey box" che restituisce soluzioni accurate simile ad ArchiGAN, utilizza un dataset molto grande per via del numero di sottomoduli da addestrare, di conseguenza richiederà considerevole tempo e risorse per restituire soluzioni soddisfacenti. Utilizzando questo approccio dobbiamo quindi affrontare un trade-off che ha da una parte l'accuratezza delle planimetrie e l'inclusione dell'utente nel processo di design, mentre dall'altra il costo computazionale dell'allenamento.

3.2 Approccio basato sui grafi

L'approccio basato sui grafi è un promettente tentativo di incorporare la teoria dei grafi per generare planimetrie e layout architettonicamente corretti. I grafi sono una modalità di rappresentazione dei dati molto versatile che mette enfasi sulle relazioni topologiche tra gli oggetti. Dal punto di vista pratico un grafo è una struttura $G(V, E)$ dove V è l'insieme dei nodi ed E rappresenta l'insieme delle coppie (n, m) con $n, m \in V$, cioè gli archi di G. Questa struttura dati risulta particolarmente adatta per modellare le planimetrie grazie alla possibilità di rappresentare informazioni o vincoli

sulla connettività delle stanze (ad esempio "la cucina deve essere connessa alla sala da pranzo" o "stanze di servizio come cabine armadio o ripostigli devono trovarsi in prossimità delle stanze da letto") e sul loro numero [9]. Questo tipo di grafo viene anche detto "grafo di layout".

La teoria sottostante a questo metodo può essere individuata nella "space syntax" [12]. Questa idea esamina come i modelli configurazionali nell'ambiente costruito sono correlati alle proprietà sociali e psicologiche dell'esperienza spaziale [6], cioè come la disposizione di elementi architettonici appartenenti all'ambiente in cui viviamo influenza gli aspetti più umani di psiche e relazioni sociali. Per raggiungere questo obiettivo la space syntax definisce due operazioni:

- riduzione e astrazione di un ambiente, nel nostro caso una planimetria, in un grafo. Cioè una serie di spazi interconnessi.
- analisi delle relazioni topologiche degli spazi usando la teoria dei grafi. [6]

La rappresentazione degli ambienti sotto forma di grafo non deve necessariamente seguire un' interpretazione geometrica, come un grafo di adiacenza delle stanze; può anche basarsi su concetti come la semantica degli spazi [12]. Possiamo quindi costruire grafi che rappresentino, ad esempio, le attività svolte in determinate stanze o sezioni dell'ambiente.

La space syntax si presta bene all'integrazione con l'intelligenza artificiale e i modelli generativi grazie alla tecnologia nota come Graph Neural Network (GNN).

3.2.1 GNN

Le Graph Neural Network sono state presentate per la prima volta nel 2008 con [16] come soluzione al problema dell'uso dei grafi nell'intelligenza artificiale. Tradizionalmente per processare dati sotto forma di grafi usando una rete neurale, occorre ridurre il grafo a un'altra struttura dati come ad esempio un vettore per poterlo fornire in input a modelli basati quel tipo

di rappresentazione. Questo però può causare una perdita delle informazioni topologiche sulle relazioni tra i nodi del grafo [15]. Le Graph Neural Network sono invece in grado di gestire i grafi direttamente, valorizzando le caratteristiche spaziali delle relazioni tra i nodi.

Questa forma particolare di rete neurale implementa una mappatura $\phi(G, n) \in \mathbb{R}^m$, dove G è un grafo, n è un nodo nel grafo e $\mathbb{R}^m$ è un vettore m-dimensionale [16]. L'idea alla base del modello è che ogni nodo n rappresenta un concetto, e ogni concetto è rappresentato da un insieme di informazioni e caratteristiche contenute in vettori chiamate *label*. A tutti i nodi e gli archi di G possono essere assegnate delle *label* (l) che esplicitano informazioni relative ai nodi e alle relazioni tra loro. Nel contesto del *Building Information Modeling*, le *label* possono contenere dati riguardanti la tipologia di stanze rappresentate dai nodi, la loro grandezza, forma, numero di porte, etc... che permettono di massimizzare la potenza espressiva geometrica del grafo come struttura dati. Ogni nodo n è anche caratterizzato da due ulteriori variabili: il primo prende il nome di *stato* del nodo n e viene indicato con il vettore $x_n \in \mathbb{R}^m$ [16]. Il suo compito è catturare una comprensione più ampia del nodo, considerando le sue relazioni con l'insieme dei nodi adiacenti (anche chiamato *intorno* del nodo). Il secondo è l'output o_n del nodo, che rappresenta una decisione riguardante il concetto incarnato dal nodo [16].

Lo stato e l'output di un nodo vengono determinati attraverso l'utilizzo di due funzioni: una di transizione locale f, anche chiamata *message function*, che definisce e predice le relazioni tra i nodi e una di output g che specifica come viene prodotto l'output di ogni nodo [16]. La funzione di transizione locale calcola lo stato di un nodo x partendo dalle informazioni provenienti dall'insieme dei nodi a esso adiacenti, in particolare le *label* di nodi e archi, gli stati dell'intorno e la *label* del nodo in questione. La funzione di output invece prende in considerazione solo informazioni locali, cioè lo stato del nodo n e la sua *label*. Definiamo con x, o i vettori che otteniamo raccogliendo insieme rispettivamente lo stato e l'output di tutti i nodi. È possibile affermare che se x e o sono unicamente definiti, allora esiste una mappatura

$\Phi : G \to \mathbb{R}^m$ da un grafo a un vettore m-dimensionale (o) ottenuto, come spiegato precedentemente, aggregando le caratteristiche e le relazioni tra i nodi attraverso lo stato x [16].

In sintesi, le GNN ci permettono di sfruttare al massimo l'utilizzo dei grafi nella rappresentazione di planimetrie. Nella prossima sezione vedremo un esempio di un modello che incorpora le GNN per apprendere e generare planimetrie partendo da un insieme corposo di requisiti.

3.2.2 Graph2Plan

In questa sezione verrà preso in considerazione uno strumento di produzione automatica di planimetrie sviluppato nel 2020, che si basa sulle tecnologie descritte nella sezione precedente: Graph2Plan. Questo software sfrutta i grafi e le GNN per offrire all'utente nuove modalità di specificare le condizioni della generazione. Oltre ad attendere in input il perimetro dell'edificio, come in [5] e [20], Graph2Plan dà all'utente la possibilità di specificare ulteriori vincoli di design, ad esempio il numero di stanze, la loro tipologia, le connessioni tra loro, etc...

Il processo di generazione di una planimetria si divide in 2 fasi:

1. il recupero di una lista di planimetrie esistenti che soddisfino i vincoli da un dataset

2. la generazione di una nuova planimetria basata su una o più tra quelle recuperate

Il programma recupera prima una lista ordinata di *grafi di layout* che soddisfino i vincoli e siano compatibili con il perimetro fornito in input. Questi grafi sono estratti da immagini provenienti da un dataset curato e preprocessato. La seconda fase del processo di generazione inizia nel momento in cui l'utente sceglie uno dei grafi di layout risultanti nella lista. Il software adatta il grafo al perimetro dell'edificio e lo trasforma in una planimetria completa. Questo adattamento del grafo prosegue in modo potenzialmente perpetuo, lasciando all'utente la possibilità di modificare i nodi del grafo scelto tramite

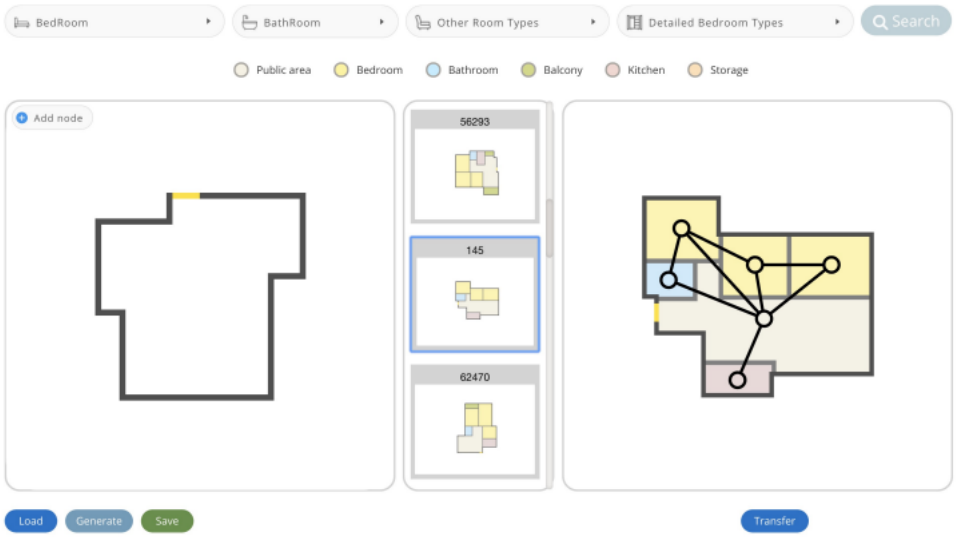


Figura 3.4: Interfaccia utente di Graph2Plan [9]

il loro spostamento, cancellazione, incremento ecc. decretandone il maggiore punto di forza del software. Graph2Plan non solo permette un'interpretazione visuale dei vincoli forniti in precedenza, ma implementa anche un design user-in-the-loop che consente di rifinire la planimetria e modificare i vincoli fino alla soddisfazione dell'utente senza dover ripetere ogni volta l'intero processo di generazione. È possibile visualizzare l'interfaccia utente del programma nella figura 3.4; a sinistra si vede il perimetro fornito dall'utente e i vincoli associati, al centro abbiamo la lista di grafi di layout rilevanti e a destra possiamo vedere la planimetria generata riadattando il grafo all'input dell'utente.

Il dataset da cui vengono estratti i grafi è RPLAN, un insieme di immagini di 80.000 planimetrie residenziali costruito da [19]. Questo campione di dati svolge sia la funzione di training set che quella di repository di planimetrie di esempio, da cui recuperare i grafi di layout che soddisfano i vincoli dell'utente. Ogni immagine nel dataset contiene delle label per tutti i pixel all'interno del perimetro, che indicano il tipo di stanza a cui appartengono quei pixel. I grafi di layout vengono estratti dalle immagini facendo uso di queste label

per rappresentare ogni stanza con un nodo e collegando ogni nodo con un arco in base alla presenza di porte o collegamenti diretti tra i diversi spazi [9].

Una volta ottenuti i grafi di layout, questi vengono forniti in input a una GNN responsabile per l'individuazione della posizione e tipologia delle stanze all'interno del perimetro. Ogni nodo del grafo i è codificato da un vettore di tre parametri: r_i, l_i, s_i. r_i è un vettore con dimensione 128 (numero scelto arbitrariamente dagli autori) che rappresenta un *embedding* della tipologia di stanza del nodo i. Esistono 13 categorie di stanze, rappresentate dalle diverse configurazioni di r_i. Il secondo dato è invece responsabile per il posizionamento della stanza i, l_i è un vettore binario di forma $\{0,1\}^{25}$. In questo modo viene rappresentata la posizione della stanza in una griglia 5×5 (da qui la dimensione del vettore) che riflette lo spazio interno al perimetro. L'ultimo parametro utilizzato per la codifica dei nodi è una rappresentazione delle dimensioni della stanza i, questo vettore s_i ha forma $\{0,1\}^{10}$ dove 10 sono le classi di grandezza di una stanza. Gli archi del grafo e_{ij} tra due nodi i, j sono codificati come vettori di dimensioni 128, volti a rappresentare gli *embedding* di 10 diversi tipi di relazioni che possono occorrere tra coppie di nodi. L'output della rete neurale è un insieme B di *bounding box* che indicano i contorni e la posizione delle stanze, ogni elemento dell'insieme ha forma $B_i = \{x_i, y_i, w_i, h_i\}$ dove i primi due dati indicano la posizione e gli altri rappresentano le dimensioni. Questo insieme viene poi fornito come input al resto dell'architettura del programma che include l'uso di una rete neurale convoluzionale per rifinire le *bounding box* e restituire una loro rappresentazione in un'immagine 128×128 di una planimetria [9].

In conclusione, Graph2Plan fa un uso efficace di grafi che contengono informazioni geometriche e spaziali per formalizzare e apprendere le regole di arrangiamento delle stanze e usarle per generare planimetrie facilmente manipolabili e modificabili dall'utente. Vedremo nella prossima sezione un esempio di utilizzo di grafi che, anziché codificare informazioni geometriche, utilizza dati sull'uso degli spazi da parte di esseri umani per generare

planimetrie sicure e funzionali.

3.2.3 Crowd simulation nella generazione con grafi

La rappresentazione di planimetrie sotto forma di dati processabili da un modello di apprendimento può basarsi sulle caratteristiche semantiche degli spazi da rappresentare. In [2] si propone un metodo di rappresentazione innovativo che si basa sulle proprietà del design degli ambienti e sulle loro interazioni con gli umani che li abitano.

La struttura dati utilizzata è sempre il grafo, che in questo caso viene integrato attraverso il graph embedding con i dati semantici della planimetria, per un'intepretazione che supera la sola geometria. Il graph embedding consiste nella trasformazione di un grafo e delle sue proprietà in un vettore di dimensioni ridotte, massimizzando la preservazione delle proprietà topologiche e informazioni contenute nel grafo [2]. Il dataset utilizzato è una versione alterata di houseExpo, chiamata houseExpo++. La versione originale consiste in 35357 rappresentazioni di planimetrie in formato JSON. Ogni rappresentazione contiene diverse informazioni riguardanti la planimetria come: il numero di stanze, la bounding box del perimetro dell'edificio e le bounding box di ogni stanza espresse in coordinate numeriche. Questi dati vengono convertiti in immagini e queste sono poi segmentate per ottenere le stanze, le loro dimensioni e le connessioni. Questo è un passaggio necessario a causa dell'inaccuratezza delle bounding box per le stanze nel dataset originale. Gli unici dati riguardanti le stanze utilizzati in houseExpo++ sono il numero e la tipologia delle stanze per etichettare gli spazi ottenuti dalla segmentazione delle immagini.

A questo punto, i dati numerici presi in considerazione finora vengono processati da *SteerSuite* (un programma di modellazione e simulazione 3D) e trasformati in modelli tridimensionali (visibile in figura 3.5). Questi modelli vengono quindi popolati con agenti virtuali in ogni stanza che hanno l'obbiettivo di lasciare l'edificio. Ciò permette il calcolo di importanti proprietà di sicurezza e accessibilità come il tempo minimo e massimo di evacuazione,

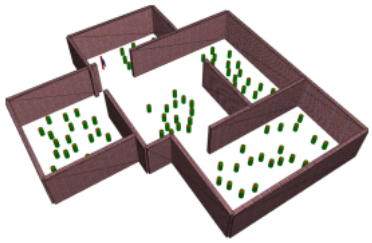


Figura 3.5: Esempio dal programma di crowd simulation SteerSuite [2]

il tasso di flusso di uscita e la distanza media percorsa dagli agenti. Queste informazioni sono inserite in vettori associati a ogni nodo del grafo. I vettori associati ai nodi contengono anche i dati sulla semantica del design delle stanze estratti precedentemente, rispettivamente le dimensioni e la tipologia della stanza. Anche gli archi del grafo vengono arricchiti con la proprietà semantica della direzione della connessione tra due stanze, rappresentata da un vettore binario di quattro dimensioni, una per ogni direzione cardinale.

Dopo aver estratto e arricchito i nodi e gli archi del grafo con attributi riguardanti la semantica e il comportamento umano, occorre rappresentare i grafi in una forma che ne rende l'analisi efficiente. Questa forma è ottenuta attraverso la mappatura da grafo a vettore (*whole-graph embedding*) utilizzando il modello *Long Short Term Memory Variational AutoEncoder* (LSTM VAE), che è solitamente usato per l'embedding di frasi o sequenze di parole. Per questo metodo è necessario prima convertire i grafi in sequenze di nodi, ottenute con la tecnica del *random walk*. Questa tecnica consiste nel scegliere un nodo di partenza e muoversi lungo un arco qualsiasi per passare al nodo successivo e ripetere il processo finché non si ottiene un cammino (sequenza di nodi) di una lunghezza prestabilita. In particolare, viene impiegata una versione di *random walk* in cui ogni arco $e_{M,N}$ ha probabilità $\frac{1}{D(N)}$ di essere scelto, dove $D(N)$ è il grado del nodo, cioè il numero delle sue connessioni [17]. Il modello viene quindi allenato utilizzando queste sequenze di nodi per ottenere una mappatura da grafo a spazio di embedding [2].

Per generare nuove planimetrie si sfrutta il concetto di *omotopia*. In uno

spazio vettoriale continuo, dati due vettori, l'omotopia è l'insieme di punti tra loro. Due vettori si dicono omotopi se uno è deformabile nell'altro in modo continuo mantenendo la linearità. La continuità dello spazio vettoriale necessaria per l'esistenza di omotopie è garantito dall'utilizzo di un VAE come modello che costruisce quello spazio [2]. Questo concetto torna utile poiché date due planimetrie, se ne può generare una nuova recuperando l'embedding al centro dell'omotopia tra i due embedding delle planimetrie di partenza. Quest'ultimo può quindi essere decodificato per ottenere il grafo corrispondente e successivamente l'immagine di una planimetria. I dati utilizzati per la rappresentazione degli ambienti influenzano in maniera fondamentale la generazione di nuove planimetrie che possiedono caratteristiche simili a entrambi i "vicini" utilizzati come input. Secondo uno user study condotto dagli autori, le planimetrie codificate con dati sulla semantica del design e interazioni ambiente-umano che risultano vicine nello spazio di embedding, sono percepite come simili dagli architetti soggetti dello studio con un'accuratezza del $77,6\%$ [2].

In conclusione, il framework proposto da questo lavoro apre nuove strade alla codifica e rappresentazione di planimetrie con l'utilizzo di dati che vanno oltre la geometria. Tuttavia, la generazione di planimetrie non è l'obbiettivo principale dello studio e il metodo presentato può essere sostituito da tecniche più complete che possono sfruttare al massimo il potenziale della rappresentazione proposta. Nella prossima sezione trarremo le conclusioni sui vantaggi dell'approccio basato sui grafi, sui contributi portati dagli studi appena visti e su possibili futuri sviluppi di queste tecnologie.

3.2.4 Conclusioni

L'approccio basato sui grafi introduce diverse tecniche promettenti alla generazione di planimetrie e presenta notevoli differenze rispetto all'approccio basato sulle immagini. Per prima cosa questo metodo permette all'utente di specificare, oltre al perimetro, dei vincoli più stringenti sulla planimetria da generare. I vincoli possono riguardare il numero di stanze che vogliamo

ottenere, la loro tipologia, la connettività tra di esse, etc... Il cambiamento di qualsiasi parametro tra questi può avere un grosso impatto sulle condizioni di vita dei potenziali abitanti dell'ambiente costruito seguendo la planimetria generata, infatti in [19] si osserva che i casi in cui gli utenti preferiscono non specificare nessun vincolo oltre al perimetro sono estremamente rari. Per queste ragioni, la facoltà di poter specificare questi requisiti risulta particolarmente vantaggiosa e distingue in modo fondamentale le soluzioni basate sulle immagini viste nelle sezioni precedenti da quelle basate sui grafi.

Questo ci porta al secondo punto a favore di questo approccio: l'incorporazione di informazioni di diversa natura nella rappresentazione delle planimetrie. Proprio grazie a questo meccanismo è possibile soddisfare i requisiti aggiuntivi di cui sopra. Infatti se i grafi associati alle planimetrie sono arricchiti con informazioni riguardanti la tipologia delle stanze, allora è possibile soddisfare requisiti riguardanti quali stanze generare. Con i grafi, le informazioni semantiche sugli spazi possono essere integrate senza soluzione di continuità con le informazioni topologiche sulla loro disposizione e geometria. Inoltre questi dati semantici possono andare a modellare diversi aspetti dell'ambiente, risolvendo problemi diversi nella generazione di planimetrie. Abbiamo visto come [2] usa informazioni sul comportamento delle folle all'interno degli edifici per ottenere planimetrie più sicure, ma un'altra strada può essere incorporare dati sulla performance energetica delle stanze all'interno della rappresentazione della planimetria. In questo modo è possibile offrire contributi non solo alla generazione di planimetrie, ma anche al calcolo dell'efficienza energetica di un'abitazione esistente partendo dalla sua planimetria, come nel caso di [1].

Riprendendo in considerazione la generazione di planimetrie tramite l'approccio basato sulle immagini, possiamo osservare un terzo punto a favore dell'utilizzo dei grafi. Infatti, il grafo come rappresentazione delle planimetrie svolge il ruolo di una struttura solida per la generazione di immagini e riduce il rischio di ottenere, tra le altre cose, stanze dalle forme poco pratiche. È difficile ottenere lo stesso risultato nell'approccio basato sulle immagini per

via della mancanza di requisiti aggiuntivi e dell'emergere molto più marcato di un bias dato dal dataset. Utilizzando i grafi, lo stile del dataset ha un impatto molto minore sulle planimetrie generate. Questo succede perché le planimetrie di allenamento vengono anch'esse trasformate in grafi e in alcuni casi come [2], la forma delle stanze non è codificata nel dataset, bensì si segue una forma standard (i.e. stanze rettangolari) stabilita a priori.

Purtroppo questa è un'arma a doppio taglio, infatti le planimetrie generate utilizzando questo approccio saranno uniformi in stile ed è molto più difficile replicare uno stile architettonico specifico, al contrario dell'approccio basato sulle immagini. Si può immaginare che il caso d'uso migliore di questo approccio sia un architetto che necessita di costruire una planimetria che segua determinati requisiti, quindi potrà generare una prima versione blanda ma funzionale usando un modello basato sui grafi, per poi rifinirlo con il proprio stile.

Il grafo è una struttura dati vantaggiosa sia dal punto di vista delle macchine, per le ragioni esposte nelle sezioni precedenti, sia per gli utenti umani grazie principalmente alla sua leggibilità e facilità di modifica. Rispetto a un'immagine la modifica di questa struttura dati è semplice e immediata, l'aggiunta di una stanza o la modifica della sua posizione è implementata con la semplice manipolazione dei nodi. Questa semplicità permette anche un'interpretazione disambigua da parte dei modelli.

In definitiva, l'uso dei grafi come base per la generazione di planimetrie rappresenta un approccio potente e versatile, in grado di soddisfare una vasta gamma di esigenze e di contribuire in modo significativo allo sviluppo e alla progettazione degli ambienti abitativi.

Capitolo 4

Proof of Concept

In questo capitolo esploreremo nella pratica alcune delle tecnologie descritte precedentemente, applicandole a uno scenario di generazione di planimetrie partendo dal perimetro dell'edificio. Lo scopo di questo esperimento è dimostrare l'accessibilità di queste tecnologie e fornire un punto di partenza per l'implementazione di un programma capace di soddisfare la domanda di ricerca. L'approccio scelto per questa proof of concept è basato sulle immagini, in particolare include l'uso di Pix2Pix, l'architettura di reti neurali che implementa una mappatura tra immagini utilizzata nei lavori di [5] e [20]. Nel nostro scenario vogliamo fornire in input al programma il perimetro di un edificio e ottenere in output una sua suddivisione in stanze. Lasciamo fuori dal processo l'arredamento delle stanze per mantenere l'output e i dati di allenamento semplici e per evitare di ottenere risultati con difetti simili a [20], che sporcano le planimetrie generate e ne inficiano la leggibilità. Questo progetto può essere considerato come un primo modulo di un software più grande. I moduli successivi potrebbero occuparsi del posizionamento dell'arredamento e della trasformazione delle immagini generate in immagini di planimetrie disegnate in uno stile più definito. Questo capitolo si dividerà in quattro sezioni: presenteremo prima i mezzi e gli strumenti utilizzati, poi verrà descritto il dataset scelto per l'allenamento e la rappresentazione delle planimetrie, successivamente verrà esposta l'architettura del modello e il

processo di allenamento per poi valutare i risultati ottenuti, infine verranno proposti futuri sviluppi e trarremo le conclusioni sull'esperimento.

4.1 Strumenti

In questa sezione verranno descritti gli strumenti utilizzati per questa *proof of concept*. L'ambiente di sviluppo scelto per la scrittura del codice e l'esecuzione dei test è Google Colab. Questa scelta è motivata dalla possibilità di utilizzare una GPU remota che ha ridotto di una quantità considerevole il tempo di allenamento rispetto all'alternativa con risorse locali. Per questo esperimento è stato necessario utilizzare un account Colab Pro per via delle frequenti interruzioni di runtime durante il processo di allenamento che sono occorse con l'uso di un account normale.

Il codice è scritto in Python con l'utilizzo di TensorFlow, una libreria open source che mette a disposizione strumenti e funzioni utili ad applicazioni di machine learning e AI. Per tracciare i grafici delle *loss function* del modello è stato utilizzato TensorBoard, il visualizzatore di TensorFlow che permette di monitorare il processo di allenamento e rappresentare diverse metriche utili.

Durante l'allenamento del modello è necessario salvare in memoria dei log per la visualizzazione dei grafici e dei checkpoint per recuperare il progresso fatto in caso di interruzione o per riprodurre uno stato particolare del modello. A questo scopo è stato usato un account Google Drive, che grazie all'integrazione con Colab rende possibile creare cartelle e salvare file direttamente da codice. Passiamo ora alla descrizione del dataset utilizzato.

4.2 Dataset

Il dataset scelto per questo esperimento è RPLAN, un insieme di 80.000 immagini di planimetrie raccolto e curato da Wu et al. [19] nel 2019. Queste planimetrie rappresentano edifici residenziali esistenti provenienti dal mercato immobiliare in Asia. Tutte le immagini nel dataset sono state filtrate per

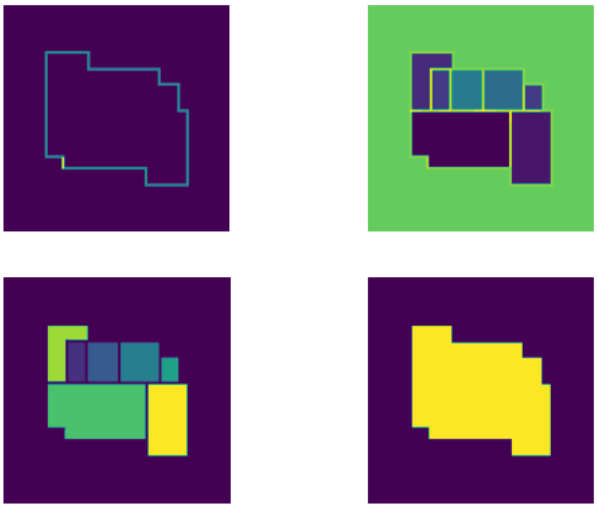


Figura 4.1: Il contenuto di ogni canale di una singola planimetria di RPLAN

evitare la presenza di stanze di tipologia non adatta alla generazione automatica come spazi per canne fumarie, ascensori, montacarichi e altri spazi troppo piccoli che possono interferire con l'apprendimento e la generazione. Tutte le planimetrie contenute in questo dataset soddisfano cinque requisiti [19]:

1. L'intera area della planimetria è compresa tra i 60 e i 120 metri quadrati.
2. Il numero totale di stanze nella piantina è compreso tra 3 e 9.
3. La planimetria include un soggiorno.
4. La proporzione tra l'area del soggiorno e l'area totale della planimetria è compresa tra 0,25 e 0,55.
5. L'area media di ogni stanza è compresa tra 10 e 20 metri quadrati.

La rappresentazione di ogni planimetria è compatta e funzionale. Tutte le immagini hanno dimensioni 256×256 e sono file png, cioè sono composte da quattro canali. Ogni canale rappresenta un aspetto diverso della planimetria, assegnando proprietà semantica a livello di pixel (4.1):

1. Il perimetro dell'edificio, dove il perimetro stesso è rappresentato da pixel con valore 127, la porta di ingresso con valore 255, e il resto ha valore 0.

2. La suddivisione in stanze con informazioni sulla tipologia (ne sono state individuate tredici A.1), ogni stanza tipologia di stanza è rappresentata con pixel di un certo valore, mentre i muri e l'esterno hanno valore 0.

3. Questo canale è usato per disambiguare stanze diverse ma con lo stesso tipo.

4. Maschera interno-esterno, l'interno è rappresentato da pixel con valore 255 mentre l'interno con valore 0.

Per permettere al modello di processare questi dati, occorre separare i canali che ci interessano in immagini distinte. Per mantenere la semplicità di input e output è stato deciso di prendere in considerazione soltanto il primo e il secondo canale, in questo modo sarà possibile fornire in input il perimetro dell'edificio (compresa la porta di ingresso) e ottenere in output la sua suddivisione in stanze di diversa tipologia. Il metodo usato per raggiungere questo obbiettivo è stato di espandere i canali interessati, ripetendoli per il numero di volte necessario per ottenere una nuova immagine che contiene solo le informazioni di un singolo canale della controparte del dataset. In questo modo da ogni immagine di RPLAN si ricavano due immagini: una di input e una di *ground truth*, necessaria per la validazione dell'output del generatore.

4.3 Modello

Il modello utilizzato in questo esperimento prende in prestito l'architettura da quello che implementa il software Pix2Pix per la mappatura tra immagini [10]. Il motivo di questa scelta è dovuto al suo utilizzo già diffuso in questo settore con buoni risultati come osservato nel capitolo precedente.

Questo modello consiste in una conditional GAN composta da un generatore basato su un'architettura a U-Net e un discriminatore rappresentato da un classificatore PatchGAN convoluzionale come proposto in [10]. Una U-Net consiste in un percorso contrattivo (lato sinistro) e in un percorso espansivo (lato destro) che insieme possono essere rappresentate da una forma a U. Il percorso contrattivo segue l'architettura tipica di una rete convoluzionale. Consiste nell'applicazione ripetuta di due convoluzioni 3x3 (con convoluzioni non accoppiate), ciascuna seguita da una ReLU e da un'operazione di max pooling 2x2 con uno stride di 2 per il downsampling. Ad ogni passaggio di downsampling raddoppiamo il numero di canali delle caratteristiche. Ogni passaggio nel percorso espansivo consiste in un upsampling della feature map seguito da una convoluzione 2x2 ("up-convolution") che dimezza il numero di canali delle caratteristiche, una concatenazione con la feature map corrispondentemente ritagliata dal percorso contrattivo e due convoluzioni 3x3, ciascuna seguita da una ReLU. Il ritaglio è necessario a causa della perdita dei pixel del bordo in ogni convoluzione. Nell'ultimo strato viene utilizzata una convoluzione 1x1 per mappare ciascun vettore di caratteristiche a 64 componenti al numero desiderato di classi. In totale, la rete ha 23 strati convoluzionali. [14]

Per ottenere risultati migliori occorre preprocessare il dataset applicando jittering casuale e mirroring alle immagini di addestramento. A questo punto siamo pronti per definire il generatore, per farlo dobbiamo implementare un encoder (il lato sinistro della U-Net) e un decoder (il lato destro). Ogni blocco all'interno dell'encoder svolgerà operazioni di convoluzione, normalizzazione del batch e leaky ReLU, una versione leggermente modificata della funzione di attivazione ReLU che per numeri negativi restituisce una retta dettata da un coefficiente predeterminato anziché restituire una curva piatta come per ReLU. Ogni blocco all'interno del decoder effettua operazioni di convoluzione trasposta, normalizzazione del batch, dropout (solo per i primi tre blocchi) e ReLU. Come per la U-Net originale sono presenti *skip connections* cioè connessioni tra feature map dal *percorso contrattivo* al *percorso espansivo*

saltando dei layer. Possiamo vedere una rappresentazione dell'architettura del generatore nella figura A.2.

Passiamo ora a definire la loss function che guiderà l'apprendimento del modello. Le cGAN apprendono secondo una loss function che penalizza le possibili strutture che differiscono dalla immagine target fornita (nel nostro caso il perimetro in input) [10]. La loss function per il generatore è la somma tra una sigmoidea *cross-entropy*, cioè una funzione che calcola la "perdita" tra l'output del discriminatore sull'immagine generata e un vettore valorizzata completamente a 1, e la funzione L1 calcolata tra l'immmagine generata e quella di target moltiplicata a un coefficiente λ. In forma compatta la loss function è:

$$S(D(x), [1, ..., 1]) + \lambda L1(x, t)$$

dove x è l'immagine generata, t è l'immagine di target e $\lambda = 100$. Il motivo per cui il primo termine dipende dall'output del discriminatore è perché vogliamo che il generatore miri a "ingannarlo", ottenendo una classificazione dell'immagine generata come reale (rappresentata dal vettore di 1). Il primo termine rappresenta effettivamente la misura di quanto il discriminatore è stato "ingannato" con successo. La scelta di questa loss function è dovuta alla sua proposizione in [10].

A questo punto possiamo descrivere il discriminatore, implementato da un classificatore convoluzionale PatchGAN. Data un'immagine, questo discriminatore tenta di classificare se una patch di dimensioni $N \times N$ dell'immagine sia reale o generata, questa operazione viene quindi ripetuta su tutta l'immagine per convoluzione e viene fatta una media dei risultati per ottenere l'output finale. Il discriminatore riceve due input: la coppia *immagine di input - immagine di target*, che dovrebbe classificare come reale e la coppia *immagine di input - immagine generata*, che dovrebbe classificare come falsa. Ogni blocco nel discriminatore effettua operazioni di convoluzione, poi normalizzazione di batch e infine leaky ReLU. Si può visualizzare l'architettura del discriminatore nella figura A.1.

La loss function del discriminatore è la somma tra la sigmoide *cross-entropy* applicata all'output del discriminatore sull'immagine reale e un vettore completamente valorizzato con 1 e la sigmoide *cross-entropy* applicata all'output del discriminatore sull'immagine generata dal generatore e un vettore completamente valorizzato con 0. Il vettore valorizzato con 1 rappresenta il comportamento che ci aspettiamo dal discriminatore sull'input dell'immagine reale, mentre il vettore valorizzato a 0 rappresenta il comportamento che ci aspettiamo nel caso dell'immagine generata. Possiamo rappresentare la loss function del discriminatore in forma compatta come segue:

$$S(D(t), [1, ..., 1]) + S(D(x), [0, ..., 0])$$

dove x rappresenta l'immagine generata e t rappresenta l'immagine di target. Vedremo nella prossima sezione come i componenti appena descritti vengono integrati per il processo di addestramento del modello.

4.4 Allenamento

Tutti questi componenti vengono integrati in uno step di allenamento, una funzione che prende in input l'immagine di un perimetro e l'immagine di target associata. Per prima cosa il generatore genera un immagine di output basandosi su quella di input, poi il discriminatore processa l'immagine di target e separatamente l'immagine generata. A questo punto vengono calcolate le loss function e i gradienti delle reti neurali, cioè la direzione secondo cui modificare le reti per minimizzare le loss function. Infine vengono applicati i gradienti al modello e vengono registrati i dati riguardanti le loss function nei log.

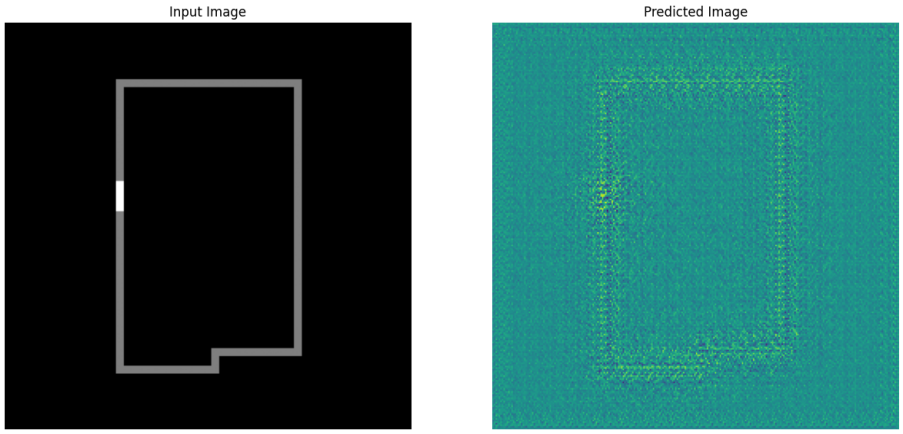
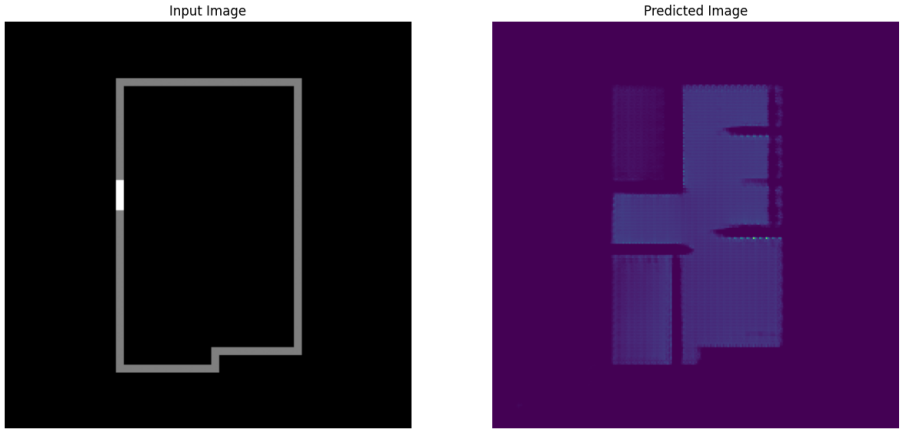
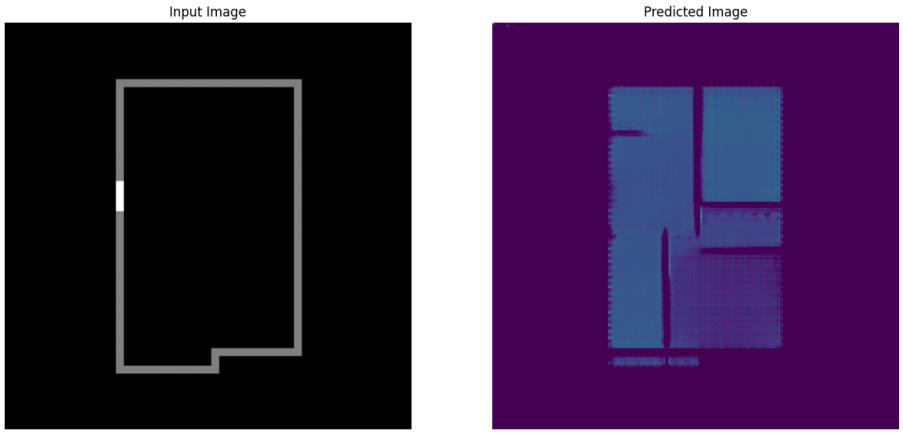
Questo processo di allenamento è stato svolto più volte con numero totale di iterazioni fissato prima a 40k, poi a 100k e infine a 60k. Il numero di iterazioni per il primo tentativo è stato scelto arbitrariamente. La qualità delle immagini in questo tentativo oscillava, spesso le immagini prodotte allo step 40k erano di difficile lettura e presentavano molti difetti e artefatti

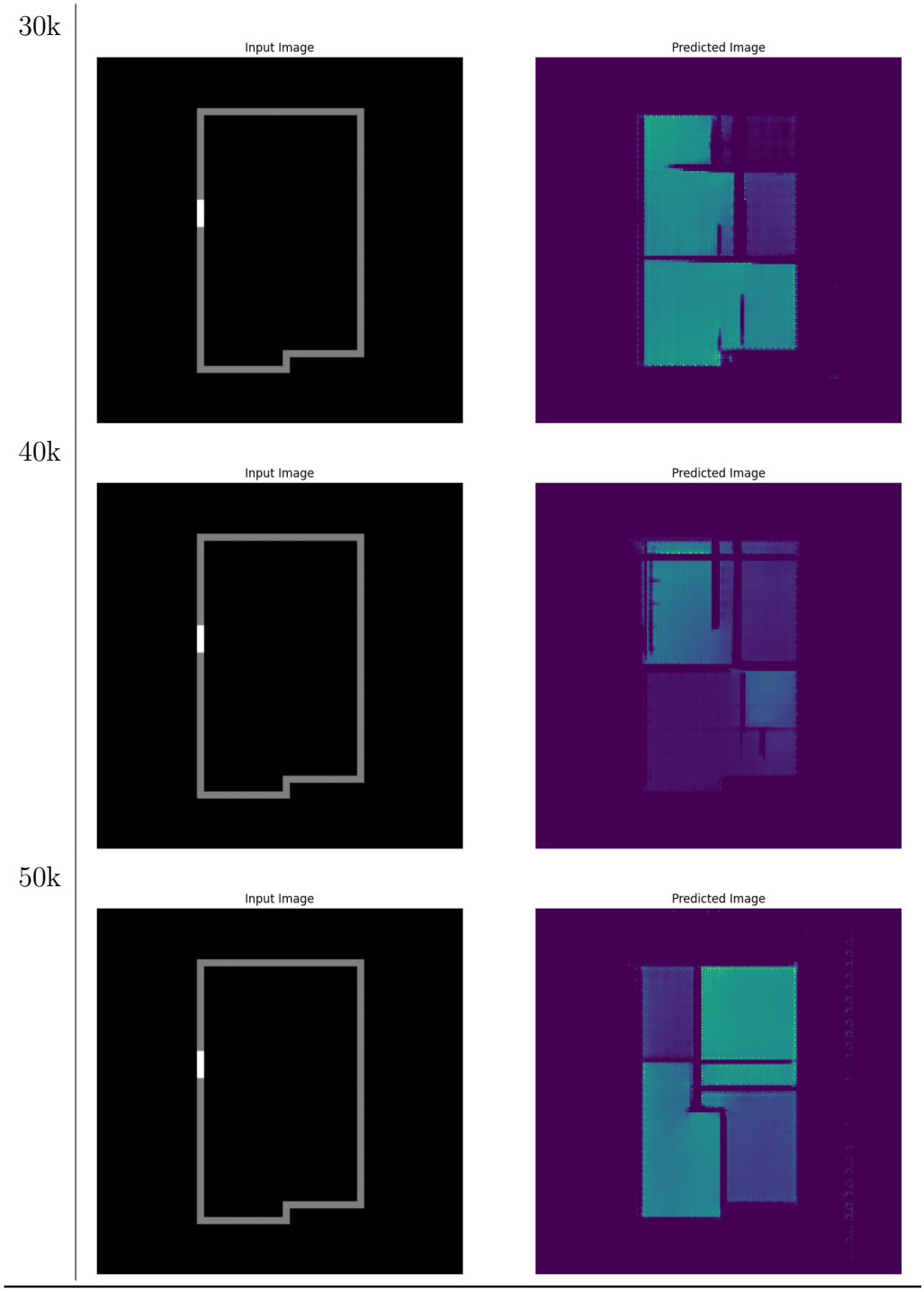
grafici. Per queste ragioni si è deciso di effettuare un nuovo tentativo aumentando il numero di iterazioni a 100k. Il processo è risultato più lineare fino all'iterazione 60k, dopo la quale si è potuto notare un notevole crollo della qualità delle immagini. Di conseguenza si è deciso di mantenere il numero di step a 60.000. L'intero addestramento ha impiegato circa 5 ore in totale, con una media di 300 secondi per mille step utilizzando una GPU NVIDIA *Tesla T4 Tensor Core*. Possiamo visualizzare nella tabella seguente lo stato del modello in vari punti dell'allenamento (4.1), ogni riga contiene un'immagine prodotta dal modello durante l'addestramento all'iterazione indicata nella prima colonna.

Possiamo osservare come il progresso dell'allenamento non sia totalmente lineare, questa impressione è supportata dall'andamento della loss function del generatore che oscilla tra un intervallo di step e l'altro pur mantenendo una tendenza generale a crescere. Tutto ciò è perfettamente in linea con il comportamento usuale che ci si aspetta da un modello GAN. Quando l'allenamento comincia (riga 0 nella tabella 4.1) il generatore produce immagini di rumore basate sul perimetro fornito. Già alle prime migliaia di iterazioni il modello è in grado di riconoscere il perimetro e isolare lo spazio interno da quello esterno. A 10k iterazioni possiamo già scorgere i primi tentativi di generare muri, che però risultano incompleti e dal posizionamento poco pratico. Il problema dei muri incompleti si ripresenterà con intensità variabile nel corso dell'allenamento e, anche nei risultati finali, sarà raro ottenere una planimetria che presenta solo muri ben tracciati. Tuttavia questo può essere mitigato in fasi successive come descritto nella sezione 4.6.

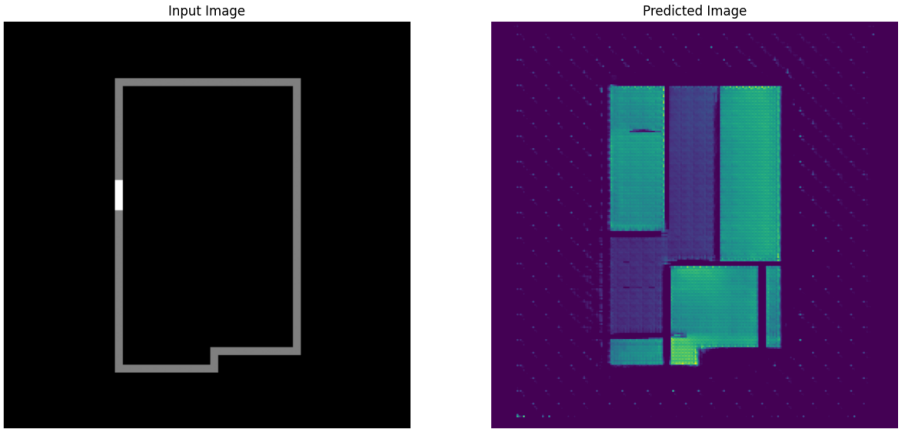
Da 20k a 30k iterazioni il modello continua a migliorare in accuratezza e genera le prime immagini con stanze ben distinte. Nelle iterazioni 30k e 40k possiamo osservare la crescita del numero di stanze, in congiunzione con il problema dei muri incompleti riscontrati nelle prime iterazioni. In queste istanze del modello spesso emergono grossi difetti come il mancato uso di spazi interni al perimetro e la generazione di stanze troppo piccole per un uso pratico. Inoltre si ripresenta il problema dei muri incompleti con intensità

Tabella 4.1: Progressione del processo di allenamento

Step	Planimetria generata
0	
10k	
20k	


30k
Input Image
Predicted Image
40k
Input Image
Predicted Image
50k
Input Image
Predicted Image

60k



discreta.

A 50k possiamo iniziare a vedere le prime planimetrie usabili. Le stanze sono ben definite, i muri sono nella maggior parte dei casi completi e la planimetria risulta semplice. La semplicità in questo tipo di immagini è intesa come il basso numero di stanze, e la prevalenza di forme meno elaborate. Questo può essere un vantaggio dato che un maggior numero di stanze porta necessariamente a stanze più piccole e con forma spesso più elaborata e meno pratica. Le planimetrie prodotte dal modello a questo punto però presentano dei difetti strutturali. Infatti in molti casi è possibile trovare delle stanze strette e lunghe, presumibilmente dei corridoi, posizionati in modo errato dal punto di vista architettonico. Questi corridoi connettono al massimo 2 o 3 stanze e sono posizionati lontani dalle porte di ingresso, dove potrebbero essere più utili e pratici.

La destinazione del processo di allenamento, rappresentata dall'iterazione 60k presenta le planimetrie migliori. Queste risultano leggibili, semplici e solide dal punto di vista semantico. Infatti riproducono spesso alcune caratteristiche architettoniche presenti nel dataset, come stanze centrali che connettono tutte le altre. Il problema dei corridoi riscontrato nelle fasi precedenti non si ripresenta qui e vi è una tendenza corretta a posizionare stanze strette e lunghe ai margini della planimetria, dove svolgono la funzione di

balconi.

Nella prossima sezione vedremo come classificare i risultati ottenuti e come possono essere comparati alle planimetrie contenute nel dataset originale.

4.5 Risultati

Per analizzare i risultati generati dal modello allenato per 60k iterazioni, dobbiamo prima fissare una metrica per la valutazione delle planimetrie prodotte. Questa metrica è implementata dalla misura di coerenza dei risultati con le planimetrie contenute nel dataset di allenamento. Possiamo dividere questa misura in quattro fattori:

1. Il rispetto delle regole di associazione tra colori e tipologia di stanze.
2. L'inclusione di una stanza che collega tutti gli spazi della planimetria.
3. Il numero di stanze deve essere compreso tra 3 e 9.
4. La semplicità della forma delle stanze e l'efficienza nell'utilizzo dello spazio fornito.

Per prima cosa notiamo che dal punto di vista grafico le planimetrie risultano ben leggibili: le forme sono chiare e il problema dei muri incompleti si presenta raramente e in misura molto minore rispetto alle planimetrie generate durante l'allenamento (4.2 terza riga, 4.3 prima riga). Tuttavia i colori, che nel dataset rappresentano il tipo di stanza (A.1), nelle immagini prodotte risultano per la maggior parte inconsistenti e di difficile lettura. Infatti se seguissimo pedissequamente la tabella dei valori associati, ci ritroveremmo con planimetrie composte da numerose stanze dello stesso tipo, con poche eccezioni. Questo ci costringe a inferire il tipo di stanza attraverso la sua forma e posizionamento. Il modello alla fine dell'allenamento sembra assegnare un solo colore correttamente e consistentemente, cioè il colore della stanza centrale che svolge il ruolo di collegamento con il resto degli spazi.

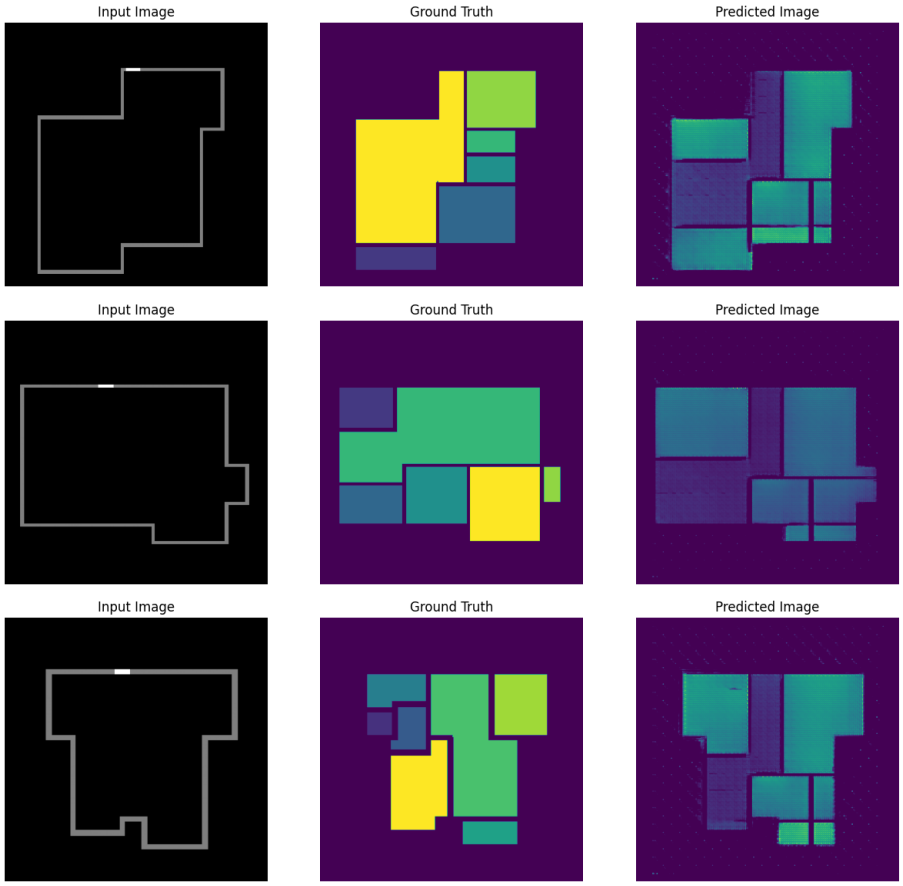


Figura 4.2: A sinistra il perimetro in input, al centro la planimetria reale associata e a destra la planimetria generata dal modello

Questo ci porta al secondo punto, tutte le planimetrie generate hanno una stanza centrale di questo tipo rappresentata negli esempi con il colore viola. In alcuni casi è possibile che non tutte le stanze siano collegate con questa area, ma questo è giustificabile quando si tratta di uno o due spazi ai margini della planimetria. Infatti nelle residenze reali è facile trovare un bagno connesso soltanto a una camera da letto o una cucina accessibile solo attraverso la sala da pranzo.

Per quanto riguarda il terzo punto, il vincolo sul numero di stanze è soddisfatto. Il modello tende a produrre planimetrie che contengono tra le 4 e le 8 stanze. Prendendo in considerazione invece il quarto punto, possiamo

notare che la configurazione delle stanze è piuttosto semplice. Le forme prevalenti sono rettangoli o la connessione di due rettangoli perpendicolari. Questo tipo di stanza risulta versatile e pratica, l'unico spazio che devia da questa forma nelle planimetrie generate è l'area centrale, che quasi sempre prende la forma visibile negli esempi (4.2).

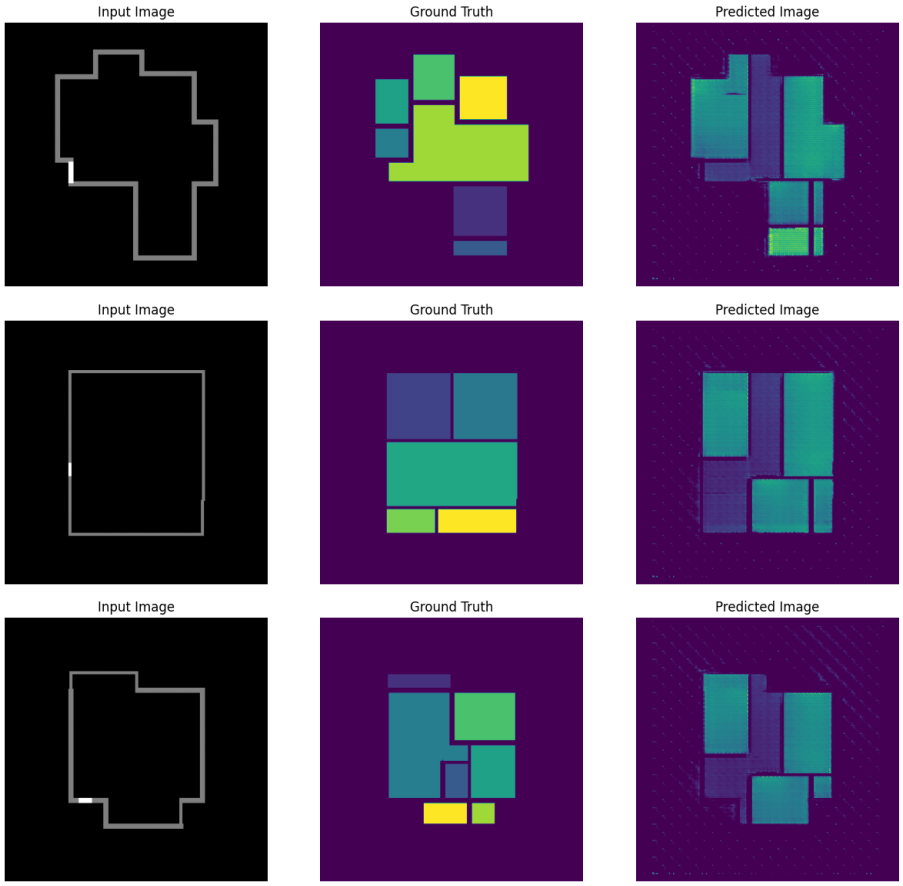


Figura 4.3: A sinistra il perimetro in input, al centro la planimetria reale associata e a destra la planimetria generata dal modello

Una limitazione evidente in molte delle planimetrie prodotte è che raramente la posizione della porta di ingresso, prefissata e presente nel perimetro in input, viene rispettata a livello di semantica degli spazi. In molte delle planimetrie prodotte la porta di ingresso, anziché dare sull'area centrale della casa, conduce in una delle stanze (i.e. 4.2, seconda riga). Questo co-

stringerebbe a riservare stanza interessata per l' atrio, quando poteva essere dedicata a una tipologia più utile e appropriata per uno spazio di quelle dimensioni, come ad esempio una camera da letto o uno studio.

Tutto sommato le planimetrie generate, a parte per la poca varietà e i difetti sopra descritti, risultano soddisfacenti e rispettano le regole dettate dal dataset. Nella prossima sezione vedremo come queste planimetrie possono essere ulteriormente sviluppate per renderle più complete e usabili dagli utenti.

4.6 Sviluppi futuri

Il modello costruito e allenato in questa proof of concept può essere considerato un sotto-modulo di un applicazione più grande. Infatti, prendendo ispirazione da [5], sarebbe possibile implementare altri moduli che partono dall'output del nostro modello e si occupano di arredare le stanze e tradurre le planimetrie da questa forma cruda a una forma disegnata più convenzionale.

Alcuni sviluppi futuri da considerare per migliorare le immagini prodotte in questo studio potrebbero riguardare l'eliminazione del problema dei muri incompleti, la standardizzazione dello spessore dei muri, la consolidazione della forma delle stanze e inferirne la tipologia attraverso la posizione e la forma.

Per risolvere il problema dei muri incompleti occorrerebbe impiegare tecniche di *image processing* per manipolare le planimetrie e decidere tra due opzioni: completare i muri generati "a metà" fino all'intersezione con un altro muro o il perimetro, oppure eliminare tutti i muri incompleti. Le stesse tecniche possono essere utilizzate per rifinire il posizionamento e lo spessore dei muri, correggendo imperfezioni come quella nella figura 4.3 terza riga, dove una parte dell'area centrale è troppo sottile per essere utilizzata e potrebbe essere assegnata alla stanza confinante.

Per inferire la tipologia delle stanze senza affidarsi ai colori, si potrebbe costruire un modello allenato su un dataset di coppie di planimetrie con stan-

ze etichettate. Questo nuovo modello dovrebbe essere in grado di prendere in input una planimetria generata dal nostro e assegnare la tipologia alle sue stanze. Questo renderebbe possibile proseguire verso l'arredamento delle stanze, data l'inevitabile dipendenza dalla loro tipologia.

4.7 Conclusioni

L'esperimento appena descritto offre una panoramica dettagliata del processo di addestramento del modello cGAN per la generazione di planimetrie. Nonostante alcuni difetti e la scarsa varietà stilistica nelle immagini prodotte, i risultati sono accettabili. Possiamo dirci soddisfatti delle planimetrie prodotte in quanto rispettano buona parte delle regole stilistiche rappresentate nel dataset e le adattano con successo a forme del perimetro varie tra loro.

I casi d'uso per modelli simili al nostro prototipo sono particolari. Riprendiamo in considerazione lo scenario su cui abbiamo aperto questa tesi: un utente, che può essere specializzato (un architetto) oppure no (il proprietario di una casa), ha bisogno di generare una planimetria per un edificio. Come abbiamo osservato nel capitolo precedente, avere l'unico requisito del perimetro è irrealistico e gli utenti preferirebbero specificare come minimo dei vincoli sul numero di stanze e la loro tipologia.

Per gestire certi requisiti occorre quindi un altro tipo di approccio, diverso da quello usato in questa *proof of concept*. Ci chiediamo quindi in quale situazione può essere utile il tipo di modello descritto in questo capitolo. La risposta è che possiamo trovare un caso d'uso insolito al nostro prodotto nella simulazione virtuale di ambienti. Questo settore presenta diversi scenari in cui la generazione automatica di spazi può fornire contributi, vediamone alcuni.

Il primo esempio è lo sviluppo di gemelli digitali urbani. Un gemello digitale urbano è un modello digitale che replica una città esistente, creato allo scopo di svolgere simulazioni su viabilità, livelli di inquinamento, qualità della vita etc... Uno strumento che genera planimetrie partendo dal peri-

metro dell'edificio sarebbe utile in quanto permetterebbe di aggiungere un nuovo livello di dettaglio al progetto senza invadere la privacy degli abitanti riproducendo le loro vere residenze. Per raggiungere un grado di accuratezza ancora più alto è possibile utilizzare planimetrie che rappresentano abitazioni nella città in questione.

Un altro esempio è lo sviluppo di videogiochi. Durante il processo di costruzione del mondo digitale in cui è ambientato un videogioco, si svolgono diverse attività volte a rendere l'ambiente credibile e immersivo. Una di queste attività è la progettazione e modellazione degli spazi. Al crescere delle dimensioni del gioco, cresce proporzionalmente anche la mole di lavoro necessario per dare forma all'ambiente. Spesso al fine di risparmiare tempo e risorse si riduce il grado di dettaglio, questo può essere mitigato da un prodotto come quello implementato in questo esperimento, che genera automaticamente configurazioni di stanze adatte alla forma dell'edificio. Le planimetrie generate possono essere interpretate da altri componenti per tradurle in spazi tridimensionali, in questo modo sarebbe possibile creare degli ambienti vari e immersivi a una frazione del costo in risorse attuale e renderebbe la creazione di opere di grande scala più accessibile.

In conclusione, nonostante il modello implementato in questo capitolo non sia adatto all'uso da parte di architetti nella stessa misura in cui lo sono altri modelli costruiti con approcci diversi, è possibile trovare scenari in cui fornirebbe preziosi contributi.

Appendice A

Proof of concept

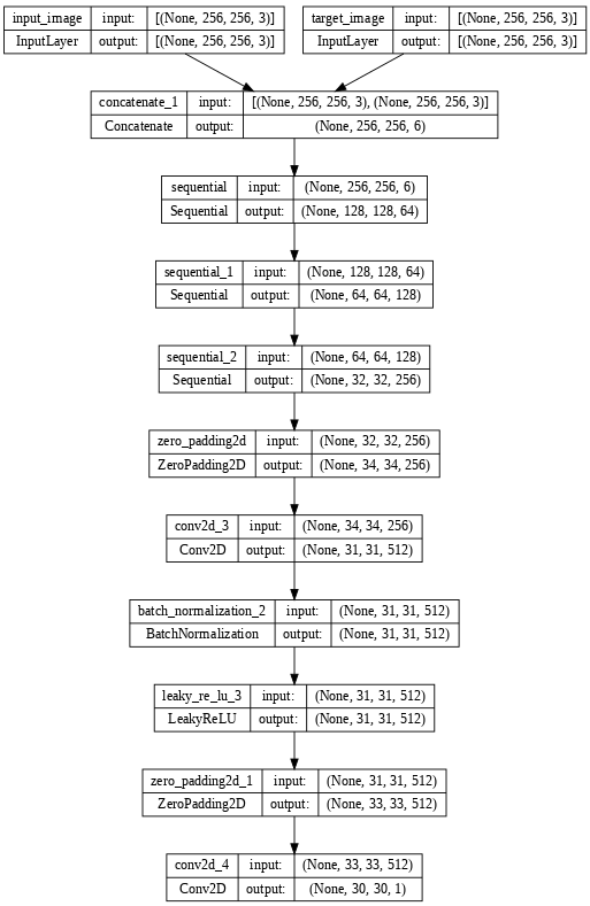


Figura A.1: Architettura del discriminatore

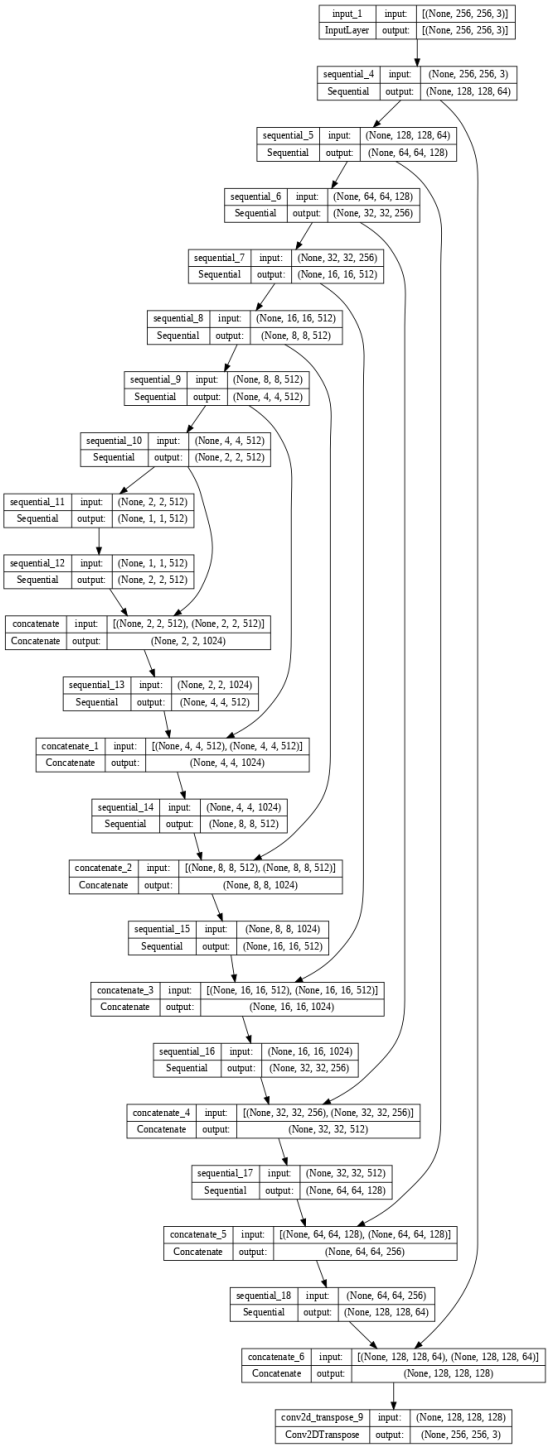


Figura A.2: Architettura del generatore

Tabella A.1: Tipologia di stanze presenti nel dataset e valore di pixel associato

soggiorno	0
camera principale	1
cucina	2
bagno	3
sala da pranzo	4
camera dei bambini	5
studio	6
seconda camera	7
camera degli ospiti	8
balcone	9
atrio	10
ripostiglio	11
wall-in	12
area esterna	13
muro esterno	14
porta di entrata	15
muro interno	16
porta interna	17

Tutto il codice utilizzato per l'esperimento è disponibile nel seguente notebook per il libero uso e consultazione.

Bibliografia

[1] Dhoyazan Al-Turki, Marios Kyriakou, Shadi Basurra, Mohamed Medhat Gaber, and Mohammed M Abdelsamea. The power of progressive active learning in floorplan images for energy assessment. 2023.

[2] Vahid Azizi, Muhammad Usman, Honglu Zhou, Petros Faloutsos, and Mubbasir Kapadia. Graph-based generative representation learning of semantically and behaviorally augmented floorplans. *The Visual Computer*, 38(8):2785–2800, 2022.

[3] Sumit Bisht, Krishnendra Shekhawat, Nitant Upasani, Rahil N Jain, Riddhesh Jayesh Tiwaskar, and Chinmay Hebbar. Transforming an adjacency graph into dimensioned floorplan layouts. In *Computer Graphics Forum*, volume 41, pages 5–22. Wiley Online Library, 2022.

[4] Silvio Carta. Self-organizing floor plans. *Harvard Data Science Review HDSR*, 2021.

[5] Stanislas Chaillou. Ai+ architecture towards a new approach. *Master's thesis. Harvard University*, 2019.

[6] Michael J Dawes and Michael J Ostwald. Applications of graph theory in architectural analysis: past, present and future research. 2013.

[7] Qianli Feng, Chenqi Guo, Fabian Benitez-Quiroz, and Aleix M Martinez. When do gans replicate? on the choice of dataset size. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6701–6710, 2021.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[9] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, 39(4):118–1, 2020.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[11] Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. Predicting the computational cost of deep learning models. In *2018 IEEE international conference on big data (Big Data)*, pages 3873–3882. IEEE, 2018.

[12] Jaechang Ko, Benjamin Ennemoser, Wonjae Yoo, Wei Yan, and Mark J Clayton. Architectural spatial layout planning using artificial intelligence. *Automation in Construction*, 154:105019, 2023.

[13] Danny Lobos and Dirk Donath. The problem of space layout in architecture: A survey and reflections. *arquiteturarevista*, 6(2):136–161, 2010.

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[15] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102, 2008.

[16] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[17] Aynaz Taheri, Kevin Gimpel, and Tanya Berger-Wolf. Learning graph representations with recurrent neural network autoencoders. *KDD Deep Learning Day*, 2018.

[18] Sean Wallish. Gan hadid. *Machine Learning and the City: Applications in Architecture and Urban Design*, pages 477–481, 2022.

[19] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019.

[20] Hao ZHENG, AN Keyao, WEI Jingxuan, and REN Yue. Apartment floor plans generation via generative adversarial networks. In *25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2020): RE: Anthropocene, Design in the Age of Humans*, pages 601–610. The Association for Computer-Aided Architectural Design Research in Asia . . . , 2020.