Università degli Studi di Modena e Reggio Emilia

DIPARTIMENTO DI INGEGNERIA "ENZO FERRARI"

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Exploring a Novel Combined Approach to Segmentation, Classification and Inpainting in Computer Vision

Candidato:
Leonardo Zini

Relatore:
Dott. Lorenzo Baraldi

ANNO ACCADEMICO 2022-2023

Declaration of Authorship

I, Leonardo Zini, declare that this thesis titled, “Exploring a Novel Combined Approach to Segmentation, Classification and Inpainting in Computer Vision” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

“And following our will and wind
we may just go where no one’s been
We’ll ride the spiral to the end
and may just go where no one’s been.”

Lateralus - TOOL

Abstract in lingua italiana

Esplorazione di un nuovo approccio combinato alla segmentazione, alla classificazione e all'inpainting nella computer vision

Leonardo Zini

L'Intelligenza Artificiale (IA), con l'uso di tecniche come il machine learning e reti neurali, sta portando una vera rivoluzione in vari settori, tra cui finanza, medicina e industria. Sta cambiando aspetti della nostra vita quotidiana e creando nuove opportunità di sviluppo. Anche il campo della Visione Artificiale sta vivendo enormi cambiamenti.

Il lavoro di tesi è stato condotto in collaborazione con l'azienda Maticad, una software house specializzata in soluzioni per il design d'interni. Si sottolinea che questo riassunto mira a fornire una panoramica generale del lavoro svolto e invita a consultare il testo completo in lingua inglese per una valutazione più dettagliata.

Il presente lavoro di tesi si basa sullo sviluppo di un sistema di visione artificiale. Si propone di fornire un quadro esaustivo dello stato dell'arte nei campi della segmentazione, classificazione ed "in-painting", con l'obiettivo finale di combinare questi elementi in un'unica pipeline, sviluppata partendo da un ambiente pre-esistente, con scopo ed esperienza dell'utente a priori ben definite . L'obiettivo principale di questo lavoro è il miglioramento qualitativo della segmentazione, attualmente basata su una rete transformer per la predizione densa, consentendo agli utenti la rimozione di oggetti, appartenenti a classi specifiche dalla scena.

Il fondamento, su cui sono stati sviluppati ulteriori elementi, è il modello "segment-anything", che si propone di segmentare qualsiasi oggetto in un'immagine in base ad un input fornito dall'utente, come una serie di punti, un rettangolo di selezione o un testo. Inizialmente, è stato adattato SAM (Segment Anything Model) alla segmentazione di regioni prefissate senza la necessità di ulteriori informazioni dall'utente, garantendo all'azienda la continuità con l'attuale interfaccia. Ciò è stato realizzato allenando un adattatore tra l'encoder dell'immagine e l'encoder dei prompt di SAM.

Parallelamente, è stato sviluppato ResnetDec, un generatore di mappe termiche, di locali interni. La dorsale del generatore è la rete convolutiva a connessioni residuali ResNet, mentre la parte di generazione è stata ideata ad-hoc per questo scopo, ed composta da una rete a convolutive trasposte con connessioni residuali.

Per la componete di rimozione dell'oggetto dalla scena sono stati analizzati diversi modelli di in-painting, basati su maschere, con il fine di selezionare quello che meglio rispettasse i vincoli - qualitativi e prestazionali - imposti.

Infine, sono stati condotti studi preliminari riguardo alla classificazione degli oggetti segmentati, sperimentando una soluzione innovativa che seguisse il paradigma "open-vocaboulary". Questo nuovo approccio si basa sull'allineamento dello spazio di codifica tra un modello di segmentazione agnostico alle classi, come segment-anything, e un modello di classificazione "open-vocaboulary" come CLIP.

In conclusione, questo lavoro di tesi esamina in dettaglio, analizza scrupolosamente e mette a confronto vari modelli all'avanguardia di diversi settori, evidenziandone i punti di forza e di debolezza. Alla luce di questa analisi completa, la tesi propone un nuovo processo di visione artificiale, presentando anche nuovi elementi. Questo nuovo processo non si limita a essere teorico, ma è progettato per essere applicato nel contesto aziendale.

Acknowledgements

First of all, I would like to deeply thank my scientific tutor and mentor, Dr. Lorenzo Baraldi, for the comparisons, advice, and guidance. I would also like to thank Maticad and its staff, Massimo, Franco, Lorenzo, and Silvia for the opportunity to develop this work and for giving it the importance it deserves. Without you, this would not have happened.

I can only be grateful and thank my family for your emotional and practical support. I know that it was not, and still is not, easy for you to fully understand me, and that makes your support all the more special. Your efforts were not entirely in vain and I will be forever grateful to you.

A key component of my journey, my colleagues first and friends later, the so-called "Cross Entropy lovers", whose name says it all. Thank you for making this journey extremely enjoyable and for the healthy and pure sense of challenge we developed for our own sake. To each and every one of you, I wish that life may offer you the best career, but most importantly, all that you deserve.

I would also like to mention my mentors, the people I have met along the path of life who have changed the way I see.

There is a group of people I want to thank, a group as diverse as it is bound by an indescribable bond, born in a neighborhood park, and which I like to think is unbreakable. Mutable, but persevering. The boys of '99. You were there when I could be nothing, and I know you will always be there, no matter where life may take us far away. And this, to me, is invaluable.

Finally, there is a group of people I have to thank. They are those people with whom I have spent days and nights, in parks and bars, people with whom I have had pure confrontations and shared visions. Your contribution has been immense. Even if we got lost with someone - that's how this game works - it doesn't mean you weren't crucial. You will always have your box in my mind. I wish you the best, wherever you go and wherever you are.

I cannot help but be grateful for what my life has been, a challenging path dotted with incredible people. I will always be thankful.

And now is the perfect time to turn our gaze toward the future.

To my brother.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and contexualization

1.1 Introduction

Artificial Intelligence (AI) is revolutionizing every facet of human endeavor, reshaping industries and challenging traditional paradigms. By mimicking cognitive functions such as learning, problem-solving, and decision-making, AI systems have emerged as powerful tools capable of transforming sectors ranging from healthcare and finance to transportation and entertainment. With its disruptive potential, AI is not merely augmenting existing processes but fundamentally redefining how businesses operate, how societies function, and how individuals interact with technology. Its ability to process vast amounts of data, uncover patterns, and make predictions has already begun to redefine efficiency, innovation, and the very nature of work across the globe.

This thesis work is a combined integration of different AI technologies applied to computer vision fields of segmentation, classification, and inpainting. The focus of this thesis is an overview of the state-of-the-art model in computer vision and their leveraging for the development of a pipeline that will have practical use in the manufacturing scenario.

This work was born in collaboration with Maticad srl, a company from Pesaro, Italy.

1.2 Maticad s.r.l.

Maticad was founded in 1991 as a software house in the CAD industry and is a company specializing in software applications for interior design with specific use of upholstery materials and home design furniture. More than 30 years in the market contribute to its established experience and know-how, with important successes and references in Italy and abroad.

1.2.1 RealityRemod

RealityRemod is one of Maticad's products, and it is an Augmented Reality Web App that can replace walls and floors covering materials directly on photographs of rooms, allowing the user to preview the color and aesthetic impact of possible new alternatives.

Reality Remod is an application that is already AI-based, and the company seeks to enhance the existing quality of outcomes while also providing users with the capability to remove objects from the scene, for two different purposes, appreciate the new coverings in the "emptied" room and replace some objects with new 3D models. The objective of this thesis is to address these two - different and complementary - company requirements.

1.3 Current Pipeline

Currently, the functions performed by Reality Remod include capturing a user-supplied image, calculating a segmentation mask to divide the image into three distinct regions (floor, wall or none), determining surface orientations, and using that information to replace them with a tile-laying algorithm, ensuring that the surfaces are replaced - with the new tiles patterns - while respecting perspective and distance, to provide the user with the most realistic experience possible. Since this product is sold primarily to tile manufacturers, and not directly to customers, each one of them will upload the necessary information for replacing

surfaces with their product. The following are the components that currently compose Reality Remod to create a clear view of the subject and better understand what has been done. Technical details will be presented in subsequent subsections.

1.3.1 Segmentation

The main component is the segmentation network. The current model used by Reality Remod is the Dense Prediction Transformer [7] (Section 2.2.2.1), trained on a composition of two datasets, ADE20k [4.1.2] and Hypersim [9], and then fine-tuned over a mixture set of real and synthetic data provided by Maticad. For further details, please refer to the relative section.

1.3.2 Normal estimation

Another fundamental information that needs to be computed is the estimation of the orientation of the various surfaces composing the image. It's intuitive that for replacing a surface, its orientation is extremely necessary. Maticad's Reslity Remod uses the pre-trained model for normal estimation called XT-Consistency and presented by Zamir *et al.* [10].

1.3.3 Posing tiles

Once the necessary information about the image, such as the segmentation of the regions and their orientations, are computed and the details of the tiles, like joints, textures, and their laying scheme, are provided, the final step is to apply an algorithm that visually displays the original image with the substituted surfaces, allowing the client to see the outcome. Reality Remod stands out from its competitors by placing the new tiles directly on the client-side, instead of the server side. This allows the end-user to adjust and align the tiles as per their preference. Reality Remod uses a 2D image to recreate the 3D scene and renders the updated scene directly on the user side using the newly processed information. Since Maticad's algorithm is their intellectual property and not within the scope of this thesis, it will not be further investigated.

1.4 About this thesis: a new pipeline

The method delineated in this thesis embarks on a multifaceted journey to elevate the quality of Reality Remod in various ways. At its core, this endeavor is driven by a dual objective: enhancing the precision and fidelity of region segmentation masks while unlocking the potential of object removal and substitution within the image scene.

Moreover, this work extends beyond mere segmentation refinement to embrace the transformative potential of object removal and insertion. By empowering users to remove objects from the scene (after classification) effortlessly, the methodology allows the unveiling of a new and cleaned vista. Furthermore, the capability to insert diverse 3D models of the removed objects introduces a layer of creativity and customization, enabling users to tailor the visual narrative to their preferences.

The main foundation of this work is Segment Anything [Section 2.2.2.2], a foundation model for the task of segmentation, developed by Meta AI and colloquially referred to as SAM. Segment-Anything serves as the foundation of this work, upon which various components are constructed to tailor a solution specifically for the company's requirements. Segment-Anything is chosen for its capability to segment objects across all categories within an input image, prompted by various input formats such as point sets, bounding boxes, or textual descriptions. This capability enlarges the scope of potential applications for Reality Remod, so expanding its utility.

However, directly integrating segment-anything into the existing framework of Reality Remod presents challenges, primarily due to its deviation from the current user experience, which does not necessitate any additional prompts from the user besides the image. Consequently, modifications are essential.

Several approaches are explored to adapt Segment-Anything to fit the current usage patterns, with each method elaborated upon later in this document.

To extend the use cases for Reality Remod to remove any object in the original image - that is the second goal of this work - and knowing that segment-anything provides a binary

mask of the selected object becomes imperative to assess state-of-the-art inpainting networks to determine the most suitable option that fulfills the given constraints while ensuring the quality of the resultant image.

Furthermore, there arises a necessity to classify the objects intended for removal from the image, enabling Maticad to restrict the usage of their applications to specific scenarios and eventually replace the removed object with a 3D model of the same class. This work compares various classification methods and introduces a novel approach to address this need.

Chapter 2

Theoretical foundations

2.1 Classification

In computer vision, classification is a fundamental task that plays a pivotal role in understanding and interpreting visual data. It involves categorizing objects or scenes into predefined classes based on their visual features. This task is significant in various domains, including autonomous driving, medical imaging, security surveillance, and industrial automation.

The classification process usually starts by obtaining visual data, such as images or videos, captured by cameras or sensors. These data are then preprocessed to improve their quality and extract relevant features that differentiate between different classes. These features may include color, texture, shape, spatial relationships, and more, depending on the specific application.

Machine learning algorithms, especially deep learning models, have transformed the field of computer vision, leading to significant improvements in classification accuracy and efficiency. Classification models are typically evaluated using objective metrics such as accuracy, precision, recall, and F1-score. These metrics quantify the model's ability to correctly classify instances across different classes. Techniques such as cross-validation and confusion matrix analysis can be used to assess the model's robustness and generalization capability.

Although computer vision classification has made significant advancements, it still presents several challenges. These include dealing with complex and cluttered backgrounds, handling occlusions, variations in lighting conditions, and ensuring robustness to scale and orientation changes.

Despite the challenges, the classification task remains a fundamental aspect of computer vision research and applications. It drives innovations that impact diverse fields and paves the way for intelligent systems capable of understanding and interpreting the visual world with human-like accuracy and efficiency.

2.1.1 Classification categories

Supervised classification involves training a model on a labeled dataset where each example has a corresponding class label. The model learns to map input features to class labels by minimizing a predefined loss function using labeled data during training.

Semi-supervised classification methods combine labeled and unlabeled data during training to improve model performance. They use information from both types of data to create more comprehensive representations of the data distribution.

Unsupervised classification groups data into clusters based on their intrinsic properties, without using class labels. Clustering algorithms like k-means or hierarchical clustering are typically used for this purpose. K-means is a fundamental unsupervised learning algorithm for clustering data into 'k' distinct groups. It iteratively assigns data points to the nearest centroid and updates centroids based on the mean of assigned points until convergence. Widely used for its simplicity and efficiency, it's crucial to note its sensitivity to initial centroid selection and the possibility of converging to local optima, often mitigated by multiple random initializations.

Self-supervised classification methods train models to predict properties of input data without explicit class labels.

2.1.2 Architectures

2.1.2.1 Convolutional Neural Networks

A Convolutional Neural Network (CNN) [11] is a deep learning architecture designed for processing structured grid-like data, such as images. CNNs leverage a hierarchical structure of interconnected layers to learn hierarchical representations from raw pixel values. Convolutional layers are central to CNNs. They apply a convolution operation, that consist in applying a sequence of kernels (a small matrix of weights) to the input image, allowing the network to capture local patterns and spatial dependencies within the input image. CNNs can efficiently extract and abstract features from images through the use of convolutional filters, pooling layers, and non-linear activation functions. This enables them to gradually learn to recognize complex patterns and objects at multiple levels of abstraction. CNNs often use techniques such as regularization, batch normalization, and dropout to enhance model generalization and prevent overfitting. They can learn discriminative features hierarchically and perform well on computer vision tasks. As a result, CNNs are widely used for image classification, object detection, segmentation, and other visual recognition tasks.

The convolution process consists of sliding the filter over the input image and computing a weighted sum of the pixel values under the filter at each position. This weighted sum produces the output pixel value at that position in the resulting image, which is called the feature map. The filter is typically much smaller than the input image, allowing it to capture local patterns and features such as edges, textures, or shapes. The fundamental operation in this kind of architecture is called **2D Cross-Correlation**, usually referred to as 2D convolution. The typical usage of CNNs is dealing with input images that have multiple channels (e.g., RGB images). Let's extend the explanation to include this aspect:

Let I be the input image and K be the kernel. Both I and K are 2D arrays. The 2D cross-correlation operation $*$ between the input image I and the kernel K is defined as follows:

$$(I * K)[m, n] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I[i, j] \cdot K[m - i, n - j]$$

Where:

- $(I * K)[m, n]$ represents the value of the output image at position (*m*,*n*).
- $I[i, j]$ denotes the pixel value of the input image at position (*i*,*j*).
- $K[m - i, n - j]$represents the value of the kernel at position (*m-i*,*n-j*). Note that the kernel is typically flipped both horizontally and vertically before the element-wise multiplication. This operation is performed for each position (*m*,*n*) in the output image, effectively sliding the kernel over the input image. For each position *(m,n)* returns a scalar, and the resulting image is the composition of this scalar.

In CNNs, these kernel filters K are typically learned and the output size depends on the size of the kernel and if padding is added. What was just explained does not take into account the channels in the image (for RGB images there are 3). When we talk about 2D convolution on multi-channel images we mean the same operation. Specifically, each kernel filter acts on all the channels of the region over which it is acting. Ultimately, the final size of the convolution operation, as far as channels are concerned, depends only on the number of filters that are applied to the image. The resulting output image represents the filtered version of the input image, where the kernel acts as a local filter that extracts specific features from the input.

ResNet ResNet stands for Residual Networks, is a revolutionary architecture in the field of deep convolutional neural networks (CNNs) that has significantly advanced computer vision. In 2015, Kaiming He et al. [1] proposed ResNet in their seminal paper 'Deep Residual Learning for Image Recognition'. ResNet introduced a novel building block called the residual block, which has revolutionized the design and training of deep neural networks.

ResNet's core innovation is its ability to address the degradation problem that arises when training very deep networks.

This approach overcomes the vanishing gradients problem that traditionally occurs as neural networks become deeper, making optimization challenging. To achieve higher accuracy,

ResNet introduces skip connections, also known as shortcut connections or identity mappings, that allow the network to learn residual functions concerning the layer inputs, as shown in 2.1. The gradient can flow directly through the network due to the skip connections, which facilitates the training of deep architectures.

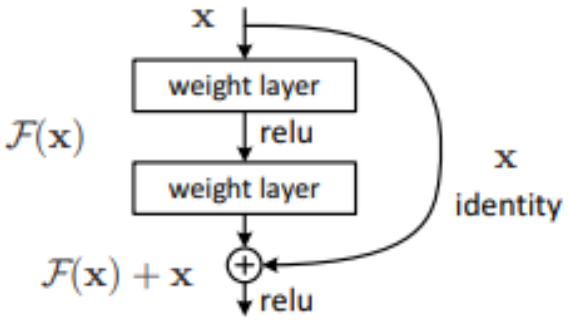


FIGURE 2.1: The Residual Block (from [1]).

ResNet's architecture is characterized by stacked residual blocks, each consisting of convolutional layers, batch normalization, and rectified linear unit (ReLU) activations. The skip connections allow the network to directly propagate information from earlier layers to deeper layers by creating shortcut paths that bypass one or more convolutional layers. This enables the network to learn residual mappings, capturing fine-grained details and facilitating the training of very deep networks with hundreds or even thousands of layers.

ResNet has shown exceptional performance in a range of computer vision tasks, such as image classification, object detection, and segmentation. It has achieved state-of-the-art results on benchmark datasets like ImageNet, surpassing previous architectures and significantly reducing error rates. Furthermore, ResNet's simple architecture and effectiveness have inspired many subsequent works and have become a cornerstone in designing deep neural networks for visual recognition tasks.

ResNet architectures exhibit variability in their configurations, primarily differing in the number of residual blocks employed, thereby influencing the total count of learnable parameters. Among the most prevalent instances are ResNet-18, ResNet-50, ResNet-101, and ResNet-152, wherein the numerical designation appended to "ResNet" denotes the number

of layers within the architectural framework. In Figure 2.2 is shown the architecture of the smaller one, ResNet-18.

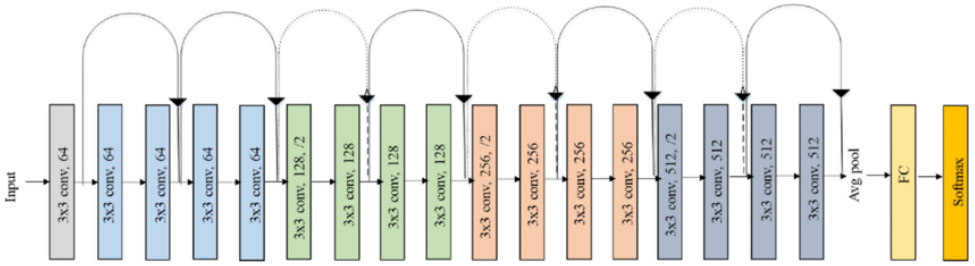


FIGURE 2.2: ResNet-18 architecture (from [2]).

2.1.2.2 Transformer

Transformers are a significant advancement in natural language processing (NLP) and have greatly impacted deep learning models for sequence processing tasks. Transformers were introduced by Vaswani et al. in 2017 in the seminal paper 'Attention is All You Need' [3]. Unlike recurrent neural networks (RNNs) and convolutional neural networks (CNNs), transformers use a self-attention mechanism to capture global dependencies and contextual information more effectively. The self-attention mechanism allows the model to weights the importance of different input tokens when generating representations.

The transformer architecture is composed of an encoder and a decoder, each consisting of multiple layers of self-attention mechanisms and feedforward neural networks. The encoder processes the input sequence during training to produce contextualized representations, while the decoder uses these representations to generate the output sequence.

Input Embeddings and Positional Encoding are the two fundamental preparatory operations. Input embedding allows to representation of the original input more densely, increasing its dimensionality but reducing the overall representation, for example, a word can be represented by a vector of dimension d_{model} (instead of a few bytes), but in the counterpart, we can represent each word in the whole dictionary by a vector of dimension d_{model}. This means that the dictionary is represented densely.

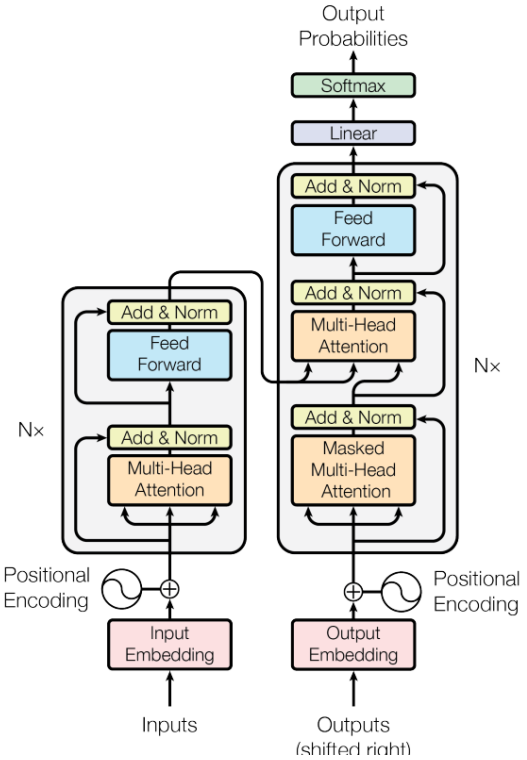


FIGURE 2.3: The Transformer Architecture (from [3]).

Position encoding is required because the transformer architecture is permutation invariant, but the information in the input is sequential, e.g., the order of words in a sentence and the order of patches in an image are essential to the overall understanding of the information.

Both input embeddings and positional encodings have dimensionality d_{model}. Typically, input embeddings are learned, while positional encoding can also be fixed. Vaswani et al. , while presenting the transformer architecture in their famous paper 'Attention Is All You Need' [3], assume a fixed positional encoding, using sine and cosine-based equations.

The Attention Operator is the foundation operator in the transformer. It takes three inputs - queries and keys of dimensions d_k and values of dimension d_v - and computes the outputs as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The operations consist of computing the dot product between a query and all the keys, dividing the result by $\sqrt{d_k}$, and then applying the softmax to get the weight of the values. This

can be done by computing the attention for each query in parallel, packing together all the queries in the matrix Q (the same is done for keys and values, respectively K and V). The scaling factor d_k is necessary since for large values of it, the dot product grows in magnitude and there may be regions after the softmax that have extremely small gradients. The attention operator just presented is the same as the one presented by the authors [Section 3.2.1 of [3]]; however, other different operators usually modify the similarity function, e.g., using cosine similarity. The first attention mechanism in each transformer decoder layer is slightly different and is called Masked Attention. Masked Attention is a variant in which we compute the attention only on known positions. For example, the prediction of position *i* depends only on known outputs in position less than *i*, ensuring the concept of sequentiality.

Multi-head Attention allows the model to attend to different parts of the input data simultaneously, enhancing its ability to capture various types of relationships and dependencies. Instead of performing the attention operation with d_{model} dimensionality, the author chose to linearly project Q, K, and V in smaller dimensions - d_k, d_k, d_v respectively. These projections are performed *h* times (where *h* is the number of heads in the multi-head attention) and each head has its own learned projection. At the end of multi-head attention, the output results are concatenated and then linearly projected again in the d_{model} dimension. Typically $d_k = d_v = d_{model}/h$, but it isn't mandatory.

Transformers have achieved remarkable success across various NLP tasks, including language modeling, machine translation, text summarization, and sentiment analysis. Models like BERT, GPT, and T5 have set new benchmarks on standard datasets and have become essential tools for NLP research and applications.

Transformers have been adapted and extended to various domains, such as computer vision, audio processing, and reinforcement learning, demonstrating their versatility and potential for advancing the state-of-the-art in machine learning tasks beyond NLP.

Vision Transformer The Vision Transformer (ViT) [4] is an innovative architecture that expands the transformer model to the field of computer vision. It provides a robust substitute for traditional convolutional neural networks (CNNs) in image comprehension tasks. In

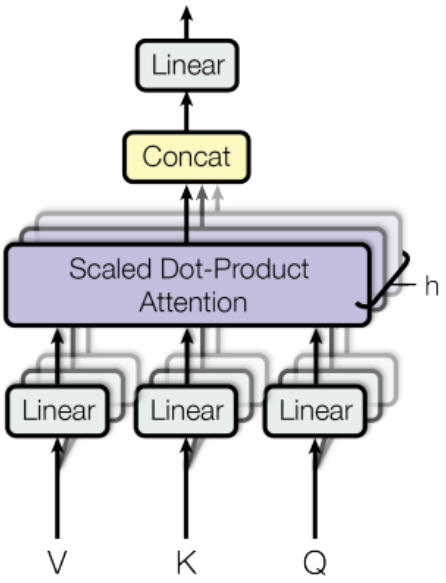


FIGURE 2.4: The Multi-Head Attention (from [3]).

2020, Dosovitskiy et al. introduced ViT in their paper 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.' ViT utilizes the self-attention mechanism of transformers to capture global dependencies and contextual information in images.

The main concept behind ViT is to represent an image as a sequence of patches. These patches are then flattened and fed into a transformer encoder. Each patch is treated as a token, similar to words in natural language processing. This allows the model to process the image holistically while preserving spatial information. ViT incorporates self-attention mechanisms, enabling the model to attend to different parts of the image and learn hierarchical representations at multiple levels of abstraction.

Applying transformers to images presents a challenge due to the large number of tokens generated by high-resolution images. To address this issue, ViT introduces patch embeddings. Each patch is linearly projected into a lower-dimensional space before being fed into the transformer encoder. This reduces the model's computational complexity and enables efficient processing of high-resolution images.

ViT incorporates positional embeddings to encode spatial relationships between patches, allowing the model to capture spatial information in images. Additionally, ViT uses multi-head self-attention to enable the model to attend to different parts of the image simultaneously, facilitating better representation learning. In summary, the Vision Transformer is a

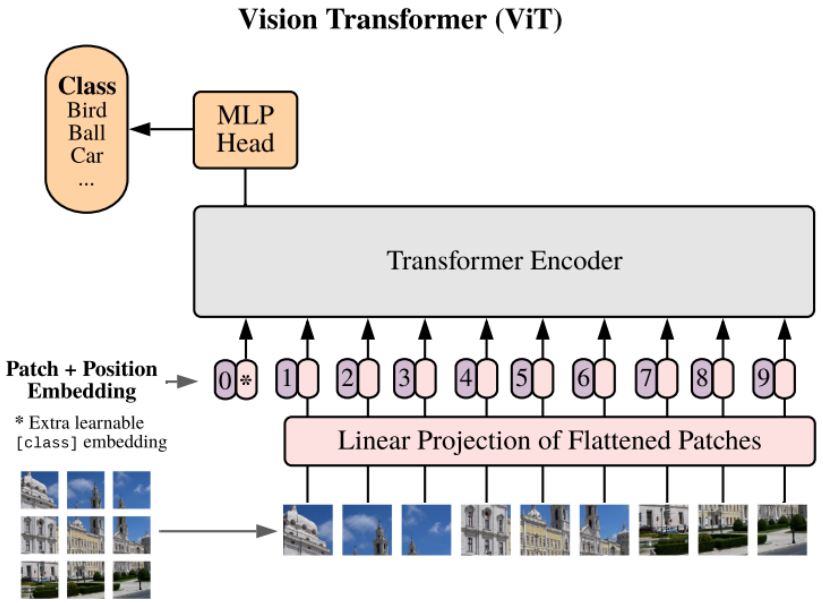


FIGURE 2.5: The Vision Transformer Architecture (from [4]).

significant advancement in computer vision, offering a flexible and scalable architecture for image-understanding tasks. By leveraging the self-attention mechanism of transformers, ViT enables models to capture global dependencies and contextual information in images, paving the way for more powerful and versatile AI systems for visual recognition.

Masked Autoencoders Masked Autoencoders are Vision transformers trained to reconstruct the original image from masked patches. It is shown in [12] that a ViT Encoder trained with this method is a multi-task learner, meaning that it can embed high to low-level information of the original image and pursue different goals. They demonstrate that transfer performance in downstream tasks outperforms supervised training [12].

2.1.2.3 Large Visual Model

Large visual models are deep learning architectures used for complex computer vision tasks, such as image classification, object detection, and segmentation. They are characterized by their size, depth, and parameter count, which allow them to learn expressive representations from vast amounts of visual data. The use of large visual models has greatly improved the state-of-the-art in computer vision, resulting in remarkable performance on benchmark datasets and driving innovation in various real-world applications.

The evolution of these models can be traced back to pioneering architectures such as AlexNet, VGG, and ResNet, which demonstrated the effectiveness of deep convolutional neural networks (CNNs) for image understanding tasks. Since then, researchers have been continuously pushing the boundaries of model size and complexity, leading to the development of increasingly sophisticated architectures capable of capturing intricate patterns and semantics in visual data.

One of the key advancements in large visual models is the incorporation of attention mechanisms, inspired by the success of transformers in natural language processing. Vision Transformer (ViT) and its variants have utilized attention mechanisms to achieve state-of-the-art results in image classification and other computer vision tasks.

Another significant trend in large visual models is the utilization of self-supervised learning and pre-training on large-scale datasets. Pre-training models on massive datasets, such as ImageNet or COCO, enables them to learn generic visual representations that can be fine-tuned for specific downstream tasks with smaller labeled datasets. This approach has proven to be highly effective in enhancing model generalization and performance across a wide range of computer vision tasks.

Training large visual models often requires significant computational resources for both training and inference due to their size and complexity. This typically involves distributed computing setups, high-performance GPUs, and sophisticated optimization techniques to handle the vast amount of data and parameters involved.

CLIP (Contrastive Language-Image Pre-training) [5] is a large visual model introduced by OpenAI in 2021 that connects vision and language understanding. Unlike traditional computer vision models that only use images, CLIP is trained on a large dataset of images paired with their corresponding textual descriptions. This allows it to learn strong visual representations based on natural language understanding.

The main innovation of CLIP is its capacity to learn image and text representations jointly in a contrastive learning framework. By aligning images and text in a shared embedding space, CLIP can associate semantically similar images and text while distinguishing between unrelated pairs. This enables the model to comprehend the relationship between visual content and textual context, facilitating a more nuanced understanding of the underlying concepts.

CLIP utilizes a transformer-based architecture, similar to those used in natural language processing tasks, to process both images and text in a unified manner.

CLIP is trained using a contrastive learning objective, where it learns to associate similar image-text pairs and distinguish dissimilar pairs, through learned embeddings. During training, positive pairs (containing matching images and text) are encouraged to have similar representations, while negative pairs (containing mismatched images and text) are encouraged to have dissimilar representations. This allows semantically similar pairs to be mapped closer together in the embedding space, while dissimilar pairs are pushed apart. As shown in 2.6, the embeddings of the text and image in the batch are computed and then compared to each other through a similarity matrix to know if the similar pairs are near also in the embedding space.

CLIP is remarkable in its ability to generalize to a wide range of downstream tasks with minimal fine-tuning. By fine-tuning CLIP on task-specific datasets with just a few examples, it can achieve competitive performance on various tasks, including image classification (as shown in Figure 2.7), object detection, and image captioning. CLIP's transferability is due to its pre-training on diverse datasets that include images and text from various domains and languages. This enables it to capture rich and generalizable visual and semantic representations.

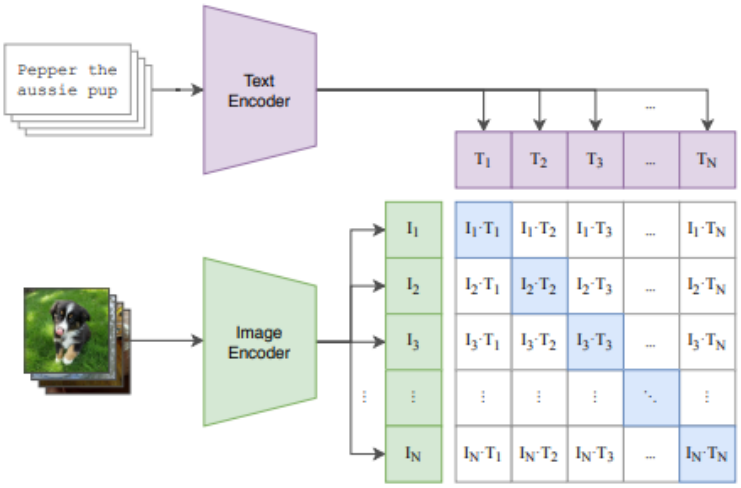


FIGURE 2.6: CLIP contrastive pretraining (from [5]).

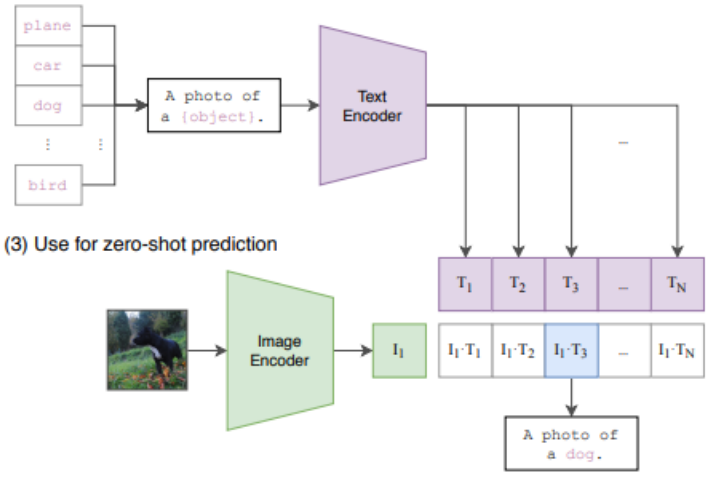


FIGURE 2.7: Example of CLIP used for zero-shot classification (from [5]), the embedding of the image is computed and then compared to the text embeddings of the class of the classification task.

CLIP has demonstrated impressive performance on benchmark datasets and has garnered significant attention in the research community for its ability to understand and reason about visual concepts through language. Its versatility and generalizability make it a valuable tool for a wide range of applications, including image understanding, content moderation, and recommendation systems.

2.2 Segmentation

Segmentation is a fundamental pillar in computer vision, playing a pivotal role in extracting rich and detailed information from visual data. Segmentation algorithms enable machines to perceive and interpret the intricate details of a scene by segmenting an image into coherent regions or segments. Unlike classification tasks, which provide a label for the entire image, segmentation offers a pixel-level understanding, carving out boundaries between objects, backgrounds, and other elements within the image. This delineation facilitates precise localization and identification of objects and allows for detailed analysis of spatial relationships, shapes, and textures present in the scene. Segmentation is a crucial aspect of various computer vision applications, such as object detection, semantic scene understanding, medical image analysis, and autonomous navigation.

2.2.1 Semantic, Instance and Panoptic Segmentation

Semantic segmentation involves assigning class labels to each pixel in an image, grouping pixels that belong to the same category or object class together. This provides a rough-level understanding of the image, where each pixel is labeled with a class such as 'car', 'person', 'building', etc.

Instance segmentation takes semantic segmentation further by not only assigning class labels to pixels but also distinguishing between individual instances of objects within the same class. This means that each pixel is not only labeled with a class but also assigned a unique instance ID, allowing for precise delineation of each object instance.

Panoptic segmentation is an advanced computer vision technique that segments both background regions and foreground objects simultaneously. This method assigns a label to every pixel in an image, indicating whether it belongs to a background region or a foreground object. The segmentation of foreground objects also includes class labels and unique instance identifiers. By combining semantic and instance segmentation, panoptic segmentation provides a comprehensive representation of scenes, capturing both the overall context and the specific details of objects within the scene

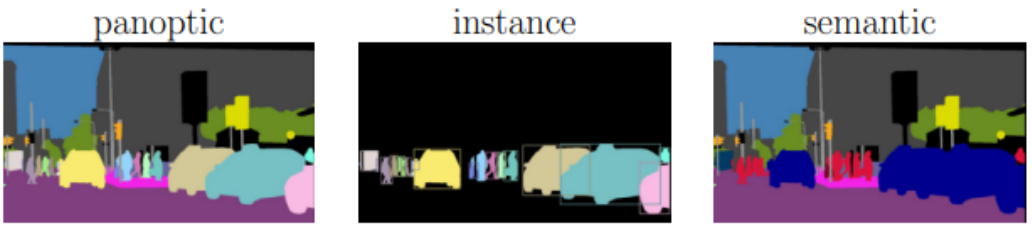


FIGURE 2.8: Segmentation tasks (from [6]).

2.2.2 Architectures

The starting point of segmentation's architectures usually are those for classification. One of the reasons is that the task of segmentation could be seen as a classification task but pixel-wise. In the realm of segmentation architectures, two distinct categories often referenced are denoted as "closed-vocabulary" and "open-vocabulary". "Closed-vocabulary" architectures maintain a fixed number of classes throughout both the training and inference phases. Conversely, "open-vocabulary" architectures possess the capability to segment any object within an image, irrespective of whether it was encountered during training.

From the perspective of end-users, the primary discrepancy between open and closed vocabulary architectures lies in the input requirements. Closed-vocabulary approaches typically operate without necessitating supplementary input, whereas open-vocabulary methods usually require additional information.

2.2.2.1 Closed vocabulary

SETR The **SE**gmentation **TR**ansformer (SETR), as proposed by recent research [13], distinguishes itself from prior architectures by avoiding the conventional approach of spatial resolution reduction. While previous methods often sacrifice spatial resolution to support larger receptive fields and abstract concepts, SETR instead employs a pure transformer architecture, similar to that utilized in Vision Transformer (ViT) models (see Section 2.1.2.2), to encode image information. This departure from resolution reduction is significant, as it allows SETR to preserve fine-grained spatial details while capturing global contextual information.

Moreover, SETR introduces two distinct convolutional-based decoder strategies. The first approach utilizes the output of the transformer encoder's final stage as input and employs a sequence of four sequential convolutional layers, each one of them followed by upsampling operations, to generate the segmentation mask. Conversely, the second strategy adopts a multi-scale feature fusion technique, leveraging information from multiple stages of the encoder to construct a comprehensive segmentation map. This versatility in decoder design underscores SETR's adaptability to diverse segmentation tasks and ensures that it can effectively handle varying levels of spatial complexity in images.

DPT Dense Prediction Transformer DPT architecture, as presented in [7], integrates vision transformers as the fundamental backbone for dense prediction tasks, replacing traditional convolutional networks. By merging tokens from different stages of the vision transformer, DPT constructs image-like representations at different resolutions, which are then merged into dense predictions by a convolutional decoder. In particular, the transformer backbone operates on representations at a consistent and comparatively high resolution, providing it with a global receptive field at each stage. This feature gives the Dense Prediction Transformer the ability to provide predictions of finer granularity and greater global coherence than fully convolutional networks.

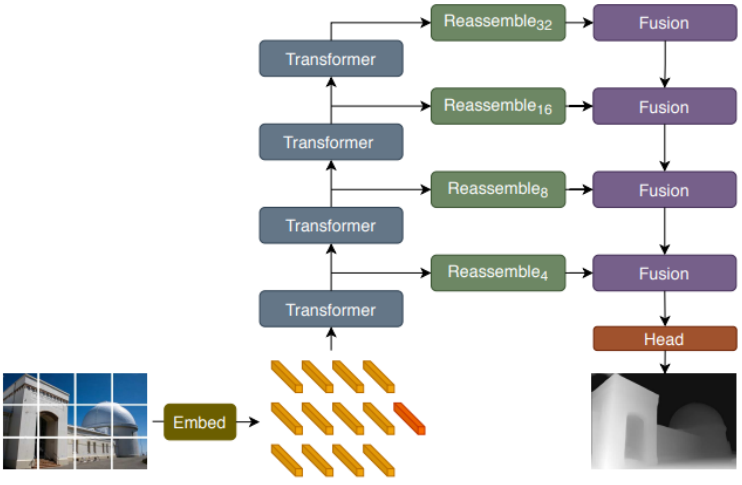


FIGURE 2.9: Dense Prediction Model Architecture (from [7]).

2.2.2.2 Open vocabulary

Segment-Anything The groundbreaking model known as Segment Anything, presented in [8] and colloquially referred to as SAM (Segment Anything Model), was introduced by Meta AI in April 2023, representing a pivotal advancement in the domain of open-vocabulary segmentation. Unlike traditional segmentation models that are limited by predefined object classes, SAM stands out for its unparalleled ability to segment virtually any object within an image. This versatility stems from its unique approach, which doesn't rely solely on direct class understanding. Instead, SAM makes its segmentation decisions based on a fine amalgamation of high and low-level information about the object itself and the context surrounding it. By leveraging this comprehensive understanding, SAM effectively transcends the constraints of conventional segmentation models, offering a more holistic and adaptable solution for image analysis tasks.

Unlike conventional models limited by predefined object categories, SAM operates within an open vocabulary framework, requiring additional contextual information to segment objects within images accurately. To achieve this, SAM integrates external prompts alongside the image input. These prompts can be points, bounding boxes, masks, or textual descriptions, enhancing SAM's ability to discern and segment diverse objects. However, it's noteworthy that as of the composition of this document, the utilization of external textual prompts remains unavailable due to the unreleased weights. Despite this limitation, SAM's incorporation of various prompts underscores its adaptability, capacity for generalization, and potential to accomplish different segmentation tasks.

Segment Anything comprises three main components: Image Encoder, Prompt Encoder, and Mask Decoder, arranged as shown in Figure 2.10 the details of each component will be presented later in this section. Segment Anything is trained over a big segmentation dataset, SA1B [See Section 4.1.1], that contains 11 million high-quality images with a total of 1.1 billion masks.

Segment Anything is a state-of-the-art model in the field of segmentation. However, its inability to determine the class of the segmented region is a significant limitation. Various

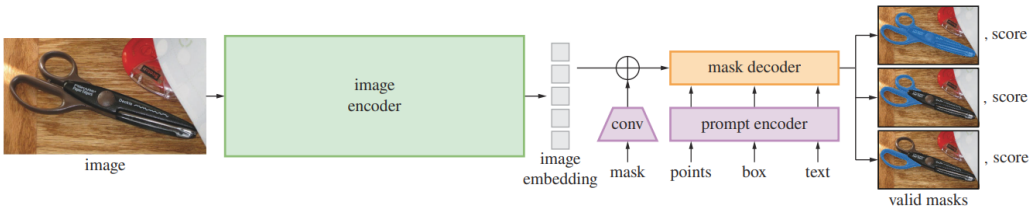


FIGURE 2.10: Segment Anything Model Architecture (from [8]).

studies have attempted to address this issue, but thus far, it remains a substantial challenge. One of the primary objectives of this thesis is to contribute to this field.

Segment Anything exists in three different versions - base, large, and huge - referring to the dimension, in number of parameters, of the image encoder.

Image Encoder The Segment Anything Image Encoder is a pre-trained Masked Autoencoder Vision Transformer adapted to process high-quality images. SAM exists in three different versions, in which the only aspect that changes is the number of parameters of the encoder. The three versions of ViT are the same as those presented in [4], ViT Base with 86 million parameters, ViT Large with 307 million parameters, and ViT Huge with 632 million.

The peculiarity of this encoder is that the image embedding is computed only once for each image, so we can use it several times for segmenting different objects without recomputing the encoding, which is the heaviest part of the process, especially from a time perspective.

Prompt Encoder Two sets of prompts are considered: sparse prompts (points, boxes, text) and dense prompts (masks). Points and boxes are represented using positional encodings, which are combined with learned embeddings specific to each prompt type. This approach allows for the incorporation of spatial and structural information into the segmentation process, enhancing the model's ability to accurately delineate objects within the image. Dense prompts, such as masks, undergo embedding through convolutions, which capture detailed features of the image regions specified by the masks. Dense embeddings will be added element-wise with the image embedding.

Mask Decoder The mask-decoder efficiently maps the image embedding, prompt embeddings, and an output token to a mask. This design means the employment of two different components, a two-way transformer decoder, which performs both the prompt-to-image attention and image-to-prompt attention in two different blocks, and a final head that maps the mask token into an actual mask (or set of masks since the decoder can predict multiple mask due to ambiguity (See 2.2.2.2). Since the output token is a concatenation of both the mask token and the Iou token, the mask decoder is also able to predict the Intersection over the Union of the generated mask(s).

Ambiguity problem The ambiguity problem within this type of architecture is non-trivial. Is difficult for segment-anything understand the semantic structure of an object. An object that needs to be segmented can be composed of several objects, for example, a man who wears a t-shirt. In this case, if SAM is fed with a single point, pointing to the t-shirt, it can't be sure if the resulting mask needs to include the person that wears it, and without context or any further information, is difficult also for a human.

Kirillov *et al.* address this task by giving in output multiple masks, representing the same semantic hierarchy of the segmented object, each of those with a predicted value of mIoU, as is possible to see in Figure 2.10.

2.2.3 Segmentation metrics

2.2.3.1 Accuracy

Adopted from the classification world, accuracy stands as a essential measure reflecting the fidelity of segmentation models in delineating object boundaries. Accuracy assesses the ratio of correctly classified pixels to the total number of pixels in the image (and it is for this reason that is inherited from classification, segmentation can be seen as pixel-wise classification), offering insights into the overall performance of segmentation algorithms. Achieving high accuracy entails effectively differentiating between foreground objects and background regions, minimizing misclassifications and ensuring precise object delineation. However,

accuracy alone may not suffice to fully capture the segmentation quality, as it does not account for imbalanced class distributions or the spatial localization of errors. Consequently, while accuracy serves as a primary metric for evaluating segmentation performance, it is often complemented by other metrics such as Intersection over Union (IoU) to provide a more comprehensive assessment, particularly in scenarios with class imbalance or intricate object boundaries.

2.2.3.2 Mean Intersection over Union

The mean Intersection over Union (mIoU) is a fundamental metric used in semantic segmentation tasks to evaluate the accuracy of segmentation models. It quantifies the overlap between predicted and ground truth segmentation masks for each class, providing a comprehensive review of segmentation performance. Calculated by dividing the intersection area of predicted and ground truth masks by their union, mIoU accounts for both false positives and false negatives, offering a balanced measure of segmentation quality. A higher mIoU value indicates a better alignment between predicted and ground truth masks, signifying superior segmentation accuracy across all classes. Widely adopted in the field of computer vision, mIoU serves as a benchmark for comparing and refining segmentation models, enabling researchers to estimate the effectiveness of different algorithms and techniques in accurately delineating object boundaries within images.

2.3 InPainting

Inpainting, within the domain of artificial intelligence (AI), represents a computational technique aimed at filling in missing or damaged regions within an image in a visually plausible manner. Stemming from the field of computer vision and image processing, inpainting algorithms have garnered significant attention due to their potential applications in various domains, including image restoration, object removal, and content creation. The fundamental objective of inpainting algorithms is to intelligently infer the missing content based on the

surrounding context and image structure, thereby seamlessly blending the inpainted regions with the existing image content.

Over the years, inpainting techniques have evolved from traditional methods, such as texture synthesis and patch-based approaches, to modern deep learning-based approaches. Deep learning-based methods have gained popularity, leveraging convolutional neural networks (CNNs) and architectures like Generative Adversarial Networks (GANs) or autoencoders. These methods learn to inpaint missing regions by training on large datasets of intact images. Recent advancements incorporate contextual attention mechanisms, allowing models to focus on relevant image regions when filling in missing information.

Inpainting finds applications in various domains including image editing, photo restoration, object removal, and medical imaging. Advancements in inpainting techniques continue to improve the quality and efficiency of inpainted results, making it a crucial component of many computer vision systems.

In the realm of inpainting methods, it is common for additional inputs beyond the image itself to be required. Typically, this additional input takes the form of a mask, delineating the specific region within the input image to be inpainted. This approach is exemplified in methods such as lama [14] and MIGAN [15], where the mask serves as a guide for the inpainting process.

However, in certain instances, textual descriptions are also provided as additional inputs beyond the mask. These textual descriptions specify how the inpainting should be carried out, often detailing aspects such as desired style, objects to be added, and other pertinent instructions. Notable examples of such methods include the Latent Diffusion Model [16].

2.3.1 Generative Adversarial Networks

Most of the inpainting methods and models are child of the Generative Adversarial Networks (GANs). GANs represent a seminal advancement in the realm of artificial intelligence, heralding a paradigm shift in generative modeling across diverse domains including images, text, and audio. Conceptualized by Ian Goodfellow et al. in [17] in 2014, GANs epitomize an

adversarial learning framework predicated upon the dynamic interaction between two neural networks: a generator and a discriminator. The generator is tasked with synthesizing data samples that approximate the underlying data distribution, while the discriminator attempts to differentiate between genuine and synthetic samples. Through iterative training, the generator refines its capacity to fabricate plausible samples, while the discriminator augments its discriminative acuity. This dialectical interplay generates a process of mutual refinement, propelling both networks toward improved performance. The essence of GANs' efficacy lies in the adversarial loss function. Adversarial loss, quantifies the discrepancy between distributions of real and generated data, compelling the generator to produce outputs indistinguishable from authentic samples and, correspondingly, minimizing the discriminator's capacity to differentiate between the two domains.

Mathematically, the adversarial loss represents a minimization of the maximum possible loss. The generator aims to maximize its deception of the discriminator, while the discriminator aims to minimize its classification error, and can be expressed as follows:

$$L_{\text{adv}}(G, D) = \mathbb{E}_{x \sim \text{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

where

- G represents the generator network.
- D denotes the discriminator network.
- $p_{\text{data}}(x)$ is the distribution of real data samples.
- $p_z(z)$ is the distribution of latent space variables.

This adversarial game between the generator and discriminator drives the network toward producing increasingly realistic and diverse data samples.

2.3.2 Inpainting metrics

The most commonly used metrics for comparing the quality of results produced by inpainting networks are the FID and LPIPS metrics. These metrics were developed specifically to evaluate images generated using GANs. Since the networks presented in this work are all based on GANs, these metrics will be used to evaluate their performance.

The **Fréchet Inception Distance (FID)** [18] is a metric commonly employed to assess the quality of generated images in Generative Adversarial Networks (GANs). It measures the similarity between the distributions of real and generated images by comparing feature representations extracted from Inception-v3, a pre-trained neural network for image classification tasks. A lower FID score suggests that the generated images closely resemble the real images in terms of their feature representations, indicating higher quality and diversity in the generated samples.

The **Learned Perceptual Image Patch Similarity (LPIPS)** is a metric used to quantify the similarity between two images based on their perceptual features. Zhang *et al.* in [19] demonstrate that using the activations of a pre-trained network (VGG on ImageNet) on two different images, measuring their distance, and aggregating it in order to obtain some similarity score for the image pair and normalizing it between 0 and 1 for an easier interpretation, can return a similarity measure more similar to what the human being perceived.

Intuitively, using pixel-wise metrics or distances can't lead to an evaluation of the perceived quality by a human, as also shown in [19].

In summary, while both LPIPS and FID are metrics used to assess image quality, LPIPS focuses on perceptual similarity at the level of image patches, while FID measures the distribution-level similarity between sets of images.

2.3.3 Mask-based Architectures

Usually, mask-based inpainting models arise from GANs, from the moment it comes to image generation, at least partially (only the areas belonging to the mask). GANs are not very

light and efficient, to generate a high-resolution image with high perceived quality requires a large amount of parameters and a conspicuous generation time, which makes this approach need improvements. The two models that will be presented have two features worth comparing.

2.3.3.1 lama

Large Mask Inpainting [14] proposes a novel approach to Inpainting, it uses Fast Fourier Convolutions (FFC), a new operator that exploits both the classical convolutional operator and the fast Fourier transform that makes the receptive field cover the entire image. The biggest model can work with images with resolutions up to 1024 x 1024 pixels without a degradation in performance also if it is trained on a dataset with images of 256 x 256 resolution, proving the robustness of the proposed model, particularly the FFC.

2.3.3.2 MiGAN

Mobile Inpainting GAN [15] is a GAN that accomplishes the inpainting task. Its main peculiarity is that it is extremely light, its main purpose - as the name suggests - is to run on mobile devices while maintaining a relatively high quality of the generated image. The author of MIGAN makes that possible thanks to adding a knowledge distillation term in the loss function during training, as explained in as explained in [20]. The network from which MIGAN distillates knowledge [21] is the Co-Mod-GAN [22]. MIGAN has been meticulously trained on datasets containing images with resolutions typically ranging between 256 x 256 and 512 x 512 pixels. This specialized training equips MIGAN with the capacity to effectively process and generate images with resolutions up to these specified maximums.

A detailed presentation of lama and MIGAN regarding aspects such as memory usage, computation time, number of parameters, and FID score will be done in Section 4.5.

2.3.4 DMs & LDMs

Briefly, Diffusion Models reach state-of-the-art performances in image synthesis through a sequential application of denoising autoencoders [16]. The cost of reaching the state-of-the-art performance is tremendously high, due to the pixel-wise operational space. To deal with this issue Rombach *et al.* presented in [16] a solution to this problem to optimize the usages of GPU in both training and inference time. The idea of Rombach *et al.* [16] is to apply DMs directly to the latent space of a trained autoencoder instead of directly in pixel space.

Also if DMs and LDMs can reach astonishing performances, they are incredibly heavy in both computational and time space. The number of parameters employed by this type of architecture is almost 20 times greater than GAN-based architecture, as enlighten by Sargsyan *et al.* in [15]. This constraint makes it absolutely impractical to use these architectures, which is why they will not be considered in the analysis of inpaiting patterns in Section 4.5.

Chapter 3

Proposed approach

In this chapter, the focus shifts towards an exploration of methodologies employed in crafting the final pipeline. The development of this pipeline represents a meticulous fusion of cutting-edge technologies, aimed at achieving superior performance and accuracy.

At the core of the methodology lies ResNetDec, a pivotal component utilized for generating heatmaps and providing crucial insights into image segmentation. Furthermore, the chapter explores the sampling algorithm inserted into the pipeline, in charge of sampling from the heatmap generated by ResNetDec. This algorithm plays a crucial role in optimizing the performance of the system since the need is to sample a few points of the same region, taking indirectly into account distances between each other and their probability.

Furthermore, integrating ResNetDec and its sampling techniques within segment-anything illustrates the dedication to maximizing the synergy between cutting-edge technologies. The integration between ResNetDec, its sampling method, and segment-anything will be named SAFLOW, meaning Segment-Anything for FLOors and Walls.

Additionally, the chapter introduces SCA (SAM-CLIP Alignment), a novel technique created to align segment-anything and CLIP, thereby giving to segment-anything the possibility to classify objects in the scene.

Each of these components will have a crucial role in the result pipeline, presented in Chapter 5.

3.1 Introducing ResNetDec

The first component introduced in this thesis, ResNetDec, derives its name from being a decoder built upon a ResNet-based backbone. Its primary objective is to generate heatmaps for indoor images based on the ResNet encoding. The structural design of the decoder's blocks aligns with the original ResNet block architecture, as depicted in Figure 2.1. However, unlike the encoding process in ResNet, which prioritizes reducing spatial resolution to augment channel dimensions, the decoder operates inversely.

The original ResNet architecture is subdivided into four layers and since preserving spatial detail is crucial for indoor scene analysis, the only initial three parts of the ResNet architecture are leveraged to extract features, enabling the preservation of higher spatial resolutions during training. ResNet was trained over images with resolutions of 224×224, while the proposed ResNetDec was almost three times greater than those typically encountered in the original ResNet setup, 768×768. The combination of using only the first three-quarters of ResNet and feeding it with images three times higher in spatial resolution leads to sufficiently preserved information in the image. In Figure 3.1, a general scheme of ResNetDec is presented.

To introduce ResNetDec, it's essential to also introduce the blocks that compose it. ResNetDec consists of two different types of blocks: a Transpose block (shown in Figure 3.2), which increases spatial resolution while decreasing the number of channels, and a Convolutional block (shown in Figure 3.3), slightly different from the original ResNet one since it always reducing the number of channels. Each block, both Transpose and Convolutional, follows the scheme of the original one, with two paths - the main one and the skip connection.

The only difference between the two types of blocks is the first layer, which is a transpose convolution in the case of a Transpose Block or a classic convolution in the case of a Convolution Block. Both blocks return a tensor with half the channels of the input, and this

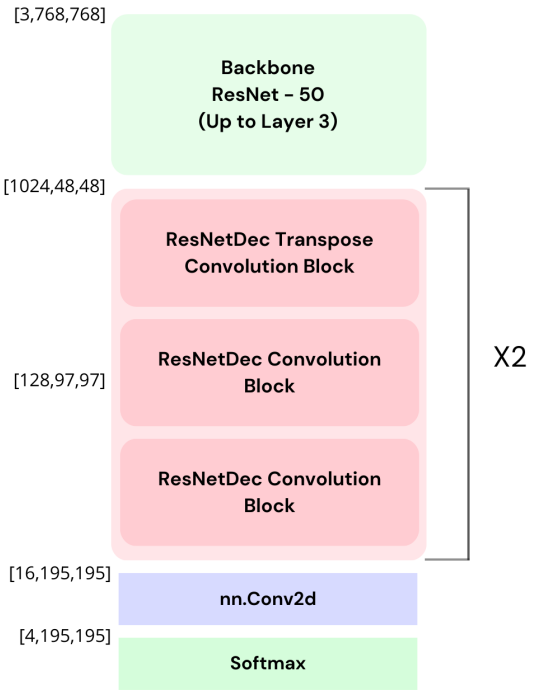


FIGURE 3.1: ResNetDec Scheme

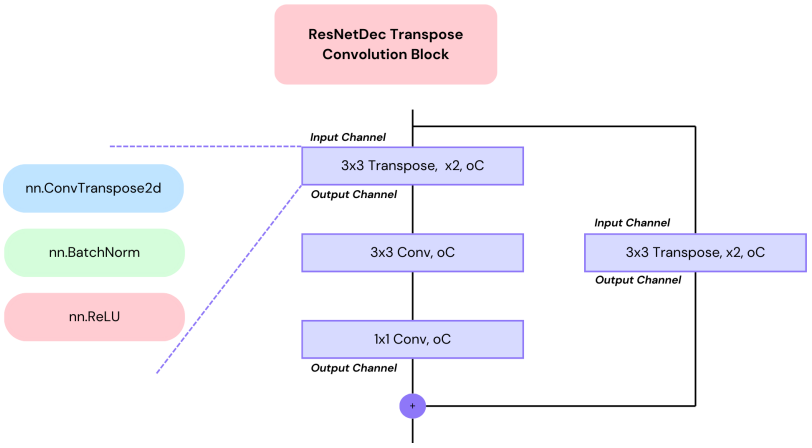


FIGURE 3.2: ResNetDec Transpose Convolutional Block

reduction is done in the first convolutional layer in both the Transpose and Convolutional blocks. The operator on the skip connection will be of the same type as the one in the first place within the block, i.e. a 1x1 2D convolution layer in Convolutional Block and a 3x3 2D transpose convolution with a stride equal to 2 in Transpose Block.

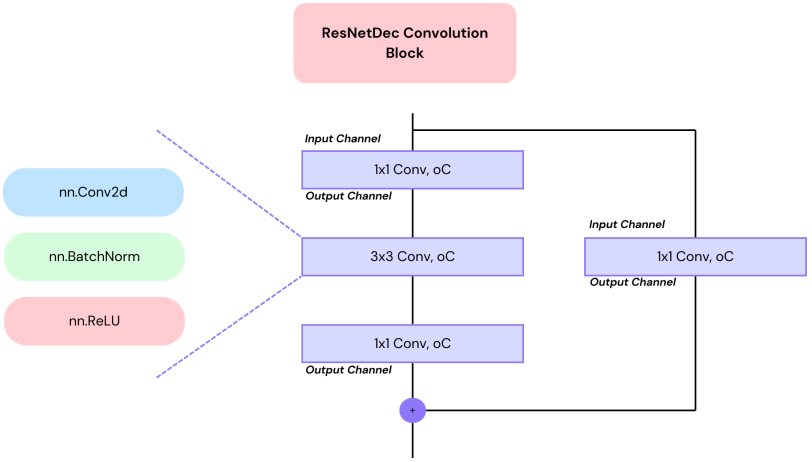


FIGURE 3.3: ResNetDec Convolutional Block

Since segmentation, as so computing a heatmap means assigning a label (or a probability) to each output pixel, the Cross-Entropy Loss has been leveraged for training ResNetDec. This component is crucial in automatizing the supplementary input prompts that the user should give to segment-anything, as already explained in Section 2.2.2.2.

3.2 Sampling techniques

Given that ResNetDec produces a heatmap delineating regions, there is the need to sample points from the ResNetDec's output in order to input it into segment-anything's prompt encoder. However, it's essential to bear in mind a crucial constraint: segment-anything is trained with up to six points in input per object. While it demonstrates robust generalization, including too many points can compromise the segmentation capacity of SAM. Therefore, careful consideration must be given to balancing the number of sampled points to ensure optimal performance. The comparison of the proposed method with other sampling algorithms will be done in Section 4.3.

3.2.1 K-means

In this presented sampling method, the process initiates with the creation of the regions' heatmap - computed for four distinct surfaces -, the definition of the total number of centroids, and the definition of the number of points per centroid.

Through the application of a max pooling convolution featuring a 2x2 kernel size, the heatmap undergoes reduction by selecting the maximum point within each region. Following this reduction, the allocation of centroids is determined based on the pixel count associated with each surface. For example, a surface encompassing 50% of the image would warrant the allocation of half of the available centroids. To maintain computational efficiency, a ceiling is imposed on the maximum number of centroids designated to any given region. Additionally, consideration of surface area ensures that centroids are not assigned to excessively small surfaces, thereby optimizing sampling efficiency and ensuring equitable representation across differing surface sizes. The centroids will be crucial information to compute the points that will be given in input to SAM to specifying the region to segment.

In addition to the outlined methodology, it's crucial to note that a quota of centroids is allocated also to the "nothing" class, representing surfaces not of interest within the heatmap. This allocation ensures that the clustering process accounts for areas devoid of significant features, maintaining a balanced representation across the entire image. As a result, not all centroids may be utilized in the procedure, reflecting the selective nature of the sampling approach.

Furthermore, to address approximation and ensure robustness in the clustering process, a minimum threshold is enforced. Regions surpassing this threshold but not substantial enough to warrant centroid assignment through the standard procedure are still allocated one centroid. This precaution guarantees that even smaller - but still big enough - regions receive some representation in the clustering analysis, enhancing the overall accuracy and completeness of the results.

Lastly, the assignment procedure is primarily governed by a proportionality principle, balancing the overall available centroids with the percentage of surface coverage within the

heatmap. This ensures a fair distribution of centroids across the image, reflecting the relative importance of each surface in the clustering analysis. By dynamically adjusting the number of centroids based on surface area, the sampling method maintains a close alignment with the underlying data distribution, optimizing the efficacy of subsequent clustering algorithms. This proportional allocation mechanism enhances the robustness and adaptability of the sampling approach, facilitating accurate analysis across diverse datasets and surface configurations.

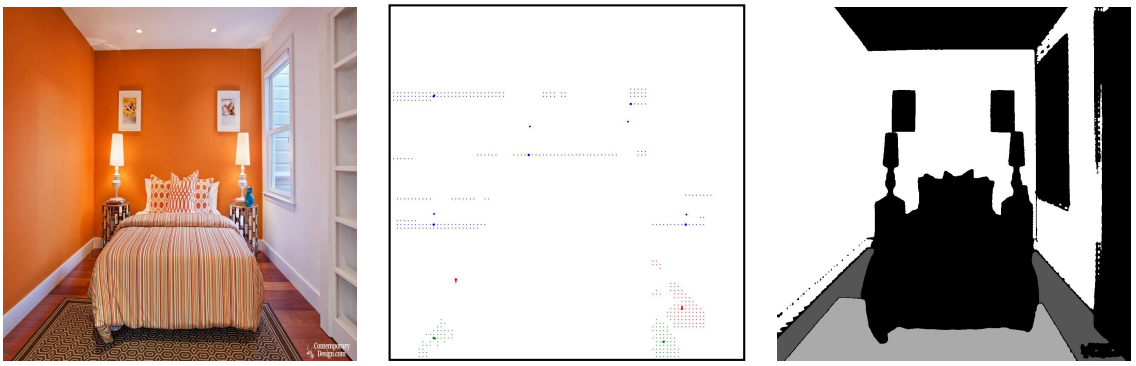
Recalling the K-Means algorithm from the theory. K-means is an iterative algorithm that partitions a dataset - in this case, a set of 2D points - into K clusters by minimizing the sum of squared distances between data points and their respective cluster centroids. This algorithm operates by iteratively assigning each data point to the nearest centroid and then recalculating the centroids based on the mean of the points assigned to each cluster.

At this point, each region will have a number of centroids, but computing its position in the image requires applying the k-means algorithm to a set of points. For each region, are taken a number of points with higher probability, and that numbee is a hyperparameter defined at the beginning of the algorithm and its meaning is the number of points that will be taken from the heatmap for every centroid associated with the region.

Moreover, the consideration of surface area plays a crucial role in this method. Larger surface areas are assigned more centroids and consequently more points for computing the centroids. This approach enables centroids to be distributed more extensively across the image, thereby capturing a richer representation of the underlying data distribution. Importantly, while centroids are distributed more widely, they still maintain a strong association with the area exhibiting the greatest probability in the original heatmap. This ensures that the centroids remain indicative of the predominant features within the image, facilitating accurate clusterization and analysis.

Finally, since centroids are not points belonging to the original set, a clustroid for each centroid is found, assuring that the final sampled points belong to the original set, to be sure that it belongs to its associated region. The clustroids is the point in the set that is nearest to its centroid.

TABLE 3.1: Mask generated from k-means approach.



The very advantage of this sampling algorithm is that is possible to define a fixed interval of points that will be given to SAM (the maximum and minimum number of centroid per region), respecting the "constraint" of not giving it too many points, and the clusterization of a much bigger set of points leads the centroids to be equally distributed in proportion to the region.

A further in-depth study of Table 3.1 will be done in Section 4.3.

3.3 SAFLOW

Segmentation of floors and walls serves as the cornerstone of this research, with its capacity for generalization being essential for the objectives outlined in this thesis. As discussed earlier in this chapter, the integration of ResNetDec is instrumental in automating the input prompt aspect of SAM, as required to be integrated to Reality Remod. Given the non-trivial nature of incorporating ResNetDec, SAFLOW (Segment Anything for FLOors and Walls) is introduced as the proposed method elucidating the interaction between ResNetDec and SAM.

In SAFLOW, we compute two distinct indoor input image embeddings: one pertaining to the native SAM's Image Encoder and the other derived from the heatmap generated by ResNetDec. Following this, a set of points for each region is sampled from the heatmap, as detailed

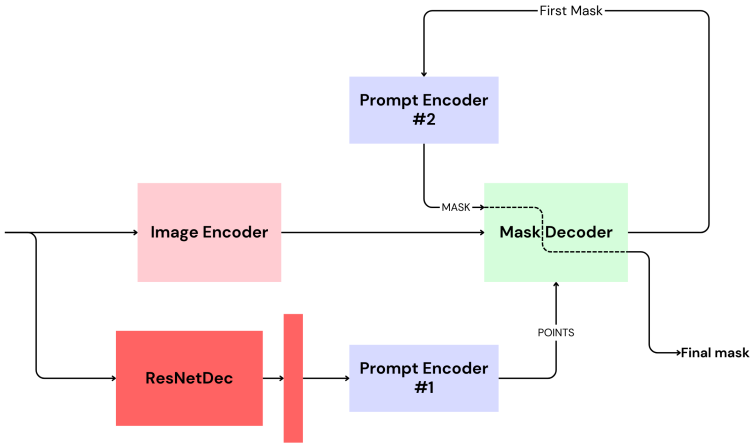


FIGURE 3.4: SAFLOW scheme.

The prompt encoder is the same, but represented two times for enlight the different inputs type.

in Section 3.2, and their embeddings are subsequently computed by the SAM's Prompt Encoder. Continuing along the native segment-anything trajectory, we proceed to derive masks for each region using the Mask Decoder.

To enhance the quality of the resultant masks further, we capitalize on SAM's Mask Decoder capability to accept a mask as input. The initial mask generated serves not as the final output, but as an additional input alongside the points sampled from ResNetDec. This iterative mask refinement approach utilizes the previous mask to improve the subsequent one.

This methodology is primarily used to solve the artifact problems described in the 4.4.3.2 section, namely the fact that SAM tends to generate dirty segmentation masks when a non-native component is used.

Figure 3.4 illustrates the logical flow of execution of the presented SAFLOW. An important thing to note is that there are two different instances of Prompt Encoder (1 and 2) in the image, but in reality there is no difference. The Prompt Encoder used will always be the same instance, but in the scheme it is shown differently to better understand the scheme and the sequence of operations.

3.4 SAM and CLIP Alignment

To remove only a precise set of classes in the image there is the necessity to classify them, although the closed-vocabulary classification is a task in which the scientific researcher has given plenty of solutions, the task of open vocabulary classification is a little bit more open. In this section, a new classification method is presented and its goal is to align the feature space of two different foundation models with different natures.

Almost every component of this work is based on segment-anything, or related to it. Despite the incredible performance in segmentation, it isn't capable of classifying the object on which it is working. Additionally, SAM (especially in the Huge version) has a strong impact on both memory usage and computational time, and this factor denies adding another heavy network to perform the classification task, like CLIP.

The work of Yuan *et al.* [23], which transfers the segmentation capacity from segment-anything to CLIP, enlightens the characteristic of both models' encoding space to embed high and low-level features from an image, demonstrating the possibility of aligning the two spaces. While Yuan *et al.* transfer the knowledge from segment-anything to CLIP, this proposed method aims to prove the feasibility of the contrary, i.e. transferring the knowledge from CLIP to segment-anything, for the reason explained above and since segment-anything, for the purpose of this work is indispensable.

3.4.1 Neck

We tried to simplify the architecture proposed by Yuan *et al.* [23] by adding a small transformer, that will act as an adapter and will be called "neck", that, given the output features vector of SAM's Image Encoder, moves it into the space of CLIP's visual embeddings. This allows the use of those features as if they were from CLIP, and so performs the classification in the same way as native CLIP does without paying the cost of having two big models loaded in memory.

The neck is a small transformer-encoder, trained first with the contrastive loss, in the same way as CLIP [5], and later with a simple Mean Square Error Loss to increase its accuracy.

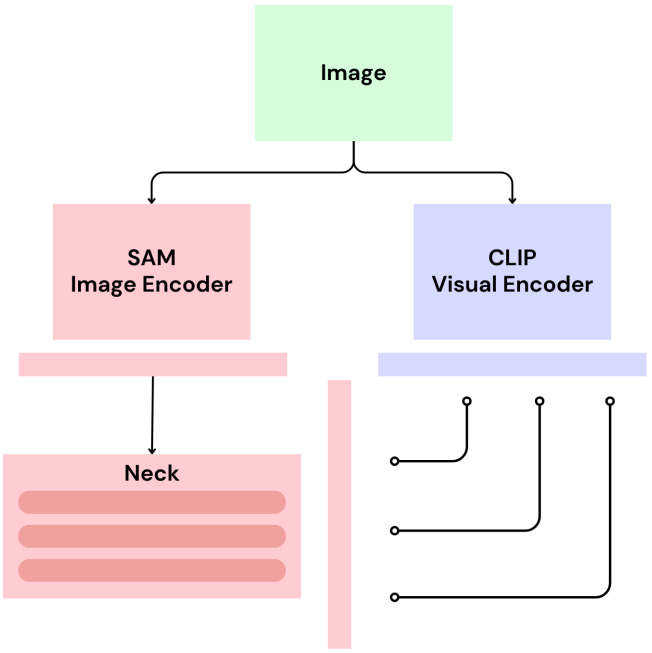


FIGURE 3.5: SAM-CLIP Alignment general scheme.

3.4.2 Classification

Once the SAM's Image Encoder features are aligned to the same space as the CLIP visual encoder, the classification is performed in the same way the original CLIP, explained in Figure 2.7. The only difference is that, in this case, the goal isn't to classify the overall but only the region defined by the mask of the object. To achieve that, the mask is leveraged to extract only the portion of the SAM's IE feature vector, align it to CLIP through the neck, and then classify it by computing the similarity between the CLIP's text embedding of the class of interest.

Chapter 4

Experiments

In this section, the experimental results are presented alongside the datasets employed. Each component is accompanied by detailed training and implementation specifics, as well as quantitative and qualitative comparisons with competitor results.

4.1 Datasets

4.1.1 SA1B

SA1B dataset [8] is the dataset over which SAM was trained. It contains 11 million high-quality and privacy-protecting images with 1.1 billion masks. The original images have a starting resolution of 3300×4950 pixels on average. Since this resolution presents a storage challenge the dataset released contains the same images but resized, bringing the shortest side of the original image to 1500 and preserving the aspect ratio. Nevertheless, the resolution of the final dataset is the higher compared to many others existing vision dataset. Since the number of images is definitely high, generating every maks manually will be a bloodbath. As states in [8] the number of automatically-annotated mask is about 99.1% . The assessment of automatic masks holds significant importance in the realm of image processing. Through a direct comparison with professional annotations and an analysis of various mask properties

against prominent segmentation datasets, the efficacy of these automated masks becomes apparent. It is evident that these automatic masks exhibit high quality and prove to be effective tools for training models in the domain of image segmentation and analysis.

4.1.2 ADE20k

The ADE20K dataset, published in 2016 by Zhou et al. [24], stands as a pivotal resource in the realm of computer vision, meticulously curated to propel the advancements in semantic segmentation and scene parsing tasks. Comprising over 20,000 medium-resolution images (20210 training images, 3000 testing images, and 2000 validation images), each intricately annotated with pixel-level semantic labels, ADE20K encapsulates a diverse array of indoor and outdoor scenes, capturing the complexities of real-world environments with remarkable fidelity. This dataset serves as a cornerstone for training and evaluating deep learning models, facilitating the development of algorithms capable of comprehensively understanding visual scenes and their constituent elements. With its rich and extensive annotations, ADE20K remains a cornerstone in fostering innovation and breakthroughs within the field of computer vision research.

The images used from ADE20k for this thesis work depict indoor scenes commonly found in residential settings, including living rooms, kitchens, bathrooms, and bedrooms, for a total of about 3420 training images and 343 validation images, to segment a few stuff classes - nothing, floors, carpets and walls.

4.1.3 Maticad indoor

Maticad Indoor presents a meticulously curated dataset tailored for the segmentation of indoor scenes within residential environments. Thus far, Maticad's focus has primarily revolved around segmenting walls and floors, reflecting the two annotated classes within the dataset. A fusion of synthetic and real images enriches the dataset, with real-life images

sourced from users of the RealityRemod service and synthetic images extracted from Maticad's 3D projects for various applications. Each image undergoes meticulous manual annotation.

The dataset carefully partitions synthetic and real images into distinct training and validation sets. However, it's worth noting the training set's inherent imbalance, comprising 1587 synthetic images and a mere 515 real images, where the final usage regards only real user-taken images. Conversely, the validation set includes 40 images from both real and synthetic images. Additionally, a test set comprising 115 unlabeled, intricate images serves as a benchmark for qualitative model comparisons.

For practical storage considerations, all images are resized, maintaining a consistent aspect ratio, with the longest side set to 1024 pixels. Real images typically exhibit a resolution of 1024×768 pixels, while synthetic counterparts span 1024×1024 pixels on average.

Since it was not created for segmenting carpet as a separate class, the segmentation mask of this dataset accorporates the mask of floor and eventually carpet into one single mask, avoiding divisions into two different classes.

4.1.4 Maticad scene

This is Maticad's second dataset, comprising 107 synthetic bathroom scenes. Each scene has a reference image and includes an annotated mask for every object depicted and its class, along with a corresponding image illustrating the scene without the specific object. On an approximated average, each scene has nine different masks with related reconstructed images, the full dataset size is 107 scenes, with an overall number of masks of 954, leading to 1061 synthetic images. The images in the datasets have high-quality resolution, but each one of them will be resized to 1024×1024 due to models constraints. An important factor to stress is the synthetic nature of this dataset, meaning that the images are not fully representative of real bathrooms, we cannot assume these data to be faithfully representative of the actual distribution, and considering also its size and variability, it isn't suitable for train networks.

4.1.5 Places2

Places2 is a comprehensive dataset introduced by Zhou et al., boasting over 10 million images capturing more than 400 distinct scenes. Each image in the dataset is standardized to a resolution of 256×256. Furthermore, Places2 comes pre-divided into train, validation, and test sets. In the training set, the number of images per class ranges from 5000 to 30,000, ensuring a diverse representation of scenes. Conversely, the validation set contains 50 images per category.

Initially curated for scene recognition tasks, Places2 has evolved over the years to become a prominent benchmark dataset for various image manipulation tasks, such as Super-Resolution or Image Inpainting. Its vast collection of diverse scenes makes it suitable for evaluating the performance of models across different visual reconstruction challenges.

By leveraging this dataset, the aim is to provide a robust assessment of inpainting algorithms, gauging their effectiveness in recreating missing or damaged portions of images concerning both perceptual similarity and statistical fidelity.

4.2 ResNetDec-based heatmap

4.2.1 Implementations and Training details

The input of ResNetDec is an RGB image with a resolution of 768x768. It enters into the backbone of the net, ResNet-50 (up to layer 3), and the output tensor, considered the embedding of the image, has 1024 channels for 48x48 spatial resolution - the starting point of ResNetDec. Intuitively,for this purposes 1024 channels with 48x48 resolution are excessive and insufficient, respectively. Therefore, ResNetDec increases spatial resolution while decreasing the number of channels gradually. This is achieved by performing two transpose convolutions at two different stages of the net, both with a stride equal to two, nearly doubling the spatial resolution each time.

At the end of ResNetDec's blocks, the output tensor has a shape of 16x195x195. Although the spatial resolution is considered sufficient, 16 channels are excessive for the goal. Therefore, a further 1x1 convolution is performed, reducing the number of channels from 16 to 4. Finally, a softmax returns the probability of each class, and consequently, the heatmap of probability for each region.

In 4.1, a detailed scheme, including changes in shape, is presented.

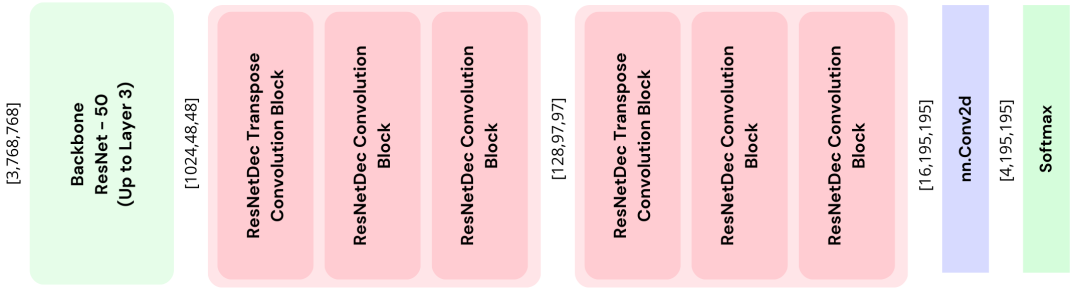


FIGURE 4.1: ResNetDec Detailed scheme

ResNetDec was exclusively trained on the ADE20k dataset using Cross-Entropy Loss with a lr of 3e-4. Given that ResNetDec's objective is to generate heatmaps highlighting floor, carpet, and wall areas within indoor images, we applied a filtering process to ADE20k in order to exclude images outside the area of interest. As a result, the dataset was refined to include 3432 indoor images, ensuring focus on relevant scenes for analysis. The same process is performed also over the validation set of ADE20k.

As a backbone, ResNet-50 is used in the version pre-trained on ImageNet-1k [25], while the weights of the decoder part were uniformly initialized. ResNet, in the version used, is trained on images with resolutions of 224×224, while in this case, the input images have a resolution of 768×768. While this aspect could be a possible problem, the first experiment done exploiting that version of ResNet has not shown any particular problems, especially since it is used as a features extractor instead of as a classificator. During the training stage, also the weights of the backbone were updated. To train ResNetDec, we used $Adam$ [26] as an optimizer, with β_1 equals to 0.9 and β_2 equals to 0.999, with a learning rate equal to 0.0003 and a batch size of 8. The training stage lasts around 4 hours on an NVIDIA GeForce RTX 2080 Ti.

4.2.2 Architecture evolution

ResNetDec must strike a delicate balance between lightweight design and precision, as it serves as a pixel-wise probability estimator. Precision is essential as any significant errors at this stage will likely propagate through the pipeline, yielding a low-quality segmentation mask. Therefore, while prioritizing lightweight architecture, maximum care is taken to ensure accuracy and precision in ResNetDec's operation. In the counterpart, the need to make ResNetDec as light as possible is to avoid excessive load on the system, especially for accomplishing this task.

As our backbone, we opted for ResNet-50 due to its ample parameter count, ensuring adequate precision in representing images. Given our intention to sample points from the resulting heatmap, it's crucial to avoid an overly compact representation that might sacrifice precision. Sampling directly from the third layer of ResNet50 implies that each point represents a window of 16x16 pixels in the original image. While this suffices for identifying general regions of interest, it falls short when pinpointing precise areas within the image portion.

To address this limitation, we need to sample points that represent as small an area as possible relative to the original image, creating a tradeoff between this and the computational load of the model. Simply put, if a point contains information about a 16x16 window and is classified by ResNetDec as belonging to a certain region, we can't be certain that its projection into the original image effectively belongs that region, because it may be misclassified by an error of approximation. To mitigate this issue, we take the feature map of ResNet-50 at three-quarters and aim to enhance its resolution as much as feasible. By doing so, we aim to preserve finer details and improve the accuracy of our heatmap-generation process, ensuring that sampled points better align with the original image's features, keeping in mind that ResNetDec must use little computational capacity and be as fast as possible.

Since we need to increase the resolution of a CNN, it is natural to leverage the 2D Transpose Convolution, which is used to increase the spatial resolution - nearly with the same

mechanism as a classical 2D Convolution. Exploiting two different 2D Transpose convolutions after the processing of the image through ResNet-50 is the baseline of the presented ResNetDec.

The initial experiment aims to enhance spatial resolution through the application of two distinct 2D transpose convolutions, each utilizing a kernel size of 3 and a stride of 2. Notably, the number of output channels is set to one-sixteenth of the input channels for both transpose convolutions. Beginning with a vector of shape [1024,48,48], as outlined earlier in this section, the application of these convolutions results in a final shape of [4,195,195]. Intriguingly, this matches the dimensions of our presented networks, such as ResNetDec, although with significantly fewer parameters. However, this reduction in parameters comes at the cost of compromised quality in the results obtained.

To augment the parameter count, we subsequently devised the Transpose Block (as described in Section 3.1), initially without incorporating skip connections, and integrated two additional convolutions following each transpose operation. In pursuit of enhanced performance, we also introduced the Convolutional Block (referenced in Section 3.1), again without incorporating skip connections at this stage.

Finally, we updated our network with a skip connection, created the definitive macro-block composed of one Transpose Block and two Convolutional Block, and tested the various models at different numbers of macro-blocks.

4.2.3 Quantitative results

In Table 4.1, we showcase the outcomes of various architectures to provide a comprehensive examination of the issue at hand.

The comparisons are conducted over two datasets: Maticad Indoor and ADE20k. Notably, ADE20k has a significantly larger collection of real images compared to Maticad Indoor. Specifically, in the validation set of indoor scenes, ADE20k comprises 343 annotated images, whereas Maticad Indoor features only 40. It's important to note that while ADE20k offers a broader dataset, its annotations are of lower quality.

Additionally, there exists a disparity in the number of classes between the two datasets. Maticad Indoor treats the carpet as part of the floor, unlike ADE20k, which distinguishes them as separate classes. At this stage, where ResNetDec must be able to, at least, detect every region if present, the performances over the ADE20k dataset are more important with respect to the one over Maticad Indoor. Also if Maticad indoor better represents the distribution of data over which ResNetDec will be used, up to now they don't cover the carpet case, so the performances over the two different datasets may have a slightly different meaning.

TABLE 4.1: ResNetDec Ablation study results. Architecture Comparisons.

	ADE20k		**Maticad Indoor**	
	mIoU	Accuracy	mIoU	Accuracy
Architectural Components				
w/ only transpose convolution	0.766	0.862	0.757	0.859
with only transpose layer	0.772	0.866	0.759	0.860
w/o Skip-connection	0.768	0.862	0.757	0.859
ResNetDec blocks				
w/ block $= 1$	0.783	0.873	0.729	0,84
w/ block $= 3$	0.784	0.874	**0.802**	**0.888**
ResNetDec (block $= 2$)	**0.787**	**0.876**	0.773	0.869

Upon scrutinizing the data presented in Table 4.1, it becomes evident that both the mean Intersection over Union (mIoU) and accuracy metrics exhibit an upward trend with the expansion of the network. It is noteworthy to observe a distinct disparity when the ResNetDec macro-block is employed. This signifies that the integration of both the Transpose Block and the Convolutional Block is crucial for achieving respectable quality outcomes.

Upon exclusive examination of the architectures incorporating the macro-block, it is observable that while the performances over ADE20k dataset are nearly proportionate, ResNetDec encounters challenges when confronted with Maticad Indoor dataset, particularly when employing a single macro-block.

Furthermore, an analysis of the performance of ResNetDec with two and three macro-blocks allows us to infer that the architecture has approached a saturation point. Given that further augmentation in the number of parameters fails to yield substantial enhancements in results,

it is prudent to adopt ResNetDec with only two macro-blocks. This choice represents a judicious compromise between quality and computational efficiency.

TABLE 4.2: ResNetDec Ablation study results. Performance Comparisons.

	Memory		**Speed**
	# of Params	Mem. usage	Inference (ms)
Architectural Components			
w/ only transpose convolution	1.19M	25.95MB	16.18
w/ only Transpose Layer	7.22M	257.08MB	17.98
w/o Skip-connection	8.46M	653.59MB	24.98
ResNetDec blocks			
w/ block $= 1$	5.455M	208.6MB	20.49
w/ block $= 3$	**17.21M**	**2123.92MB**	**27.219**
ResNetDec (block $= 2$)	13.42M	774.94MB	25.167

To fortify this deduction, an examination of Table 4.2 underscores that while the inference speed remains notably low across each approach, the memory utilization by ResNetDec employing three macro-blocks in the Forward/Backward pass markedly exceeds that of the configuration with two macro-blocks. This disproportional increase in memory consumption fails to justify the marginal improvements achieved.

Upon closer scrutiny of the metrics delineated in Table 4.1, one may contend that the optimal architecture is the one exclusively utilizing transpose convolutional layers. This assertion arises from the observation that the disparity in mIoU and accuracy metrics across both datasets is little.

To substantiate the validity of the aforementioned analysis and elucidate why ResNetDec necessitates two macro-blocks, it is imperative to take a look at the qualitative results expounded in the subsequent subsection.

4.2.4 Qualitative results

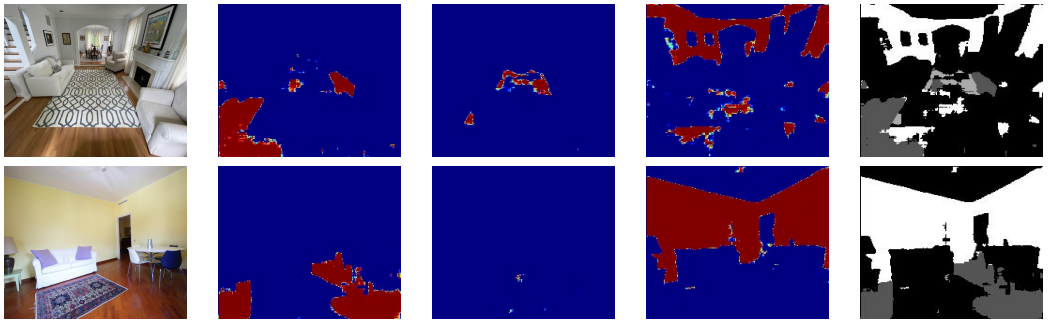
The qualitative results provided herein represent the evolution output of ResNetDec architectures. It is pertinent to reiterate that ResNetDec generates outputs of dimensions $[4, 195, 195]$

for each element in the batch, where each element corresponds to a softmax probability vector associated with each pixel. Each vector comprises four elements denoting the probability of that pixel belonging to one of the following classes: nothing, floor, carpet, or wall, respectively.

In the following tables, only the heatmaps pertaining to the classes of interest are depicted, following the aforementioned order. The tables comprise the original image, floor heatmap, carpet heatmap, wall heatmap, and segmented image. The segmented image is derived from the output of ResNetDec by associating each pixel with the class possessing the highest probability, adhering to the conventional segmentation practice.

Each heatmap is presented as an image of dimensions 195×195 (except for ResNetDec with one macro-block, where it is 97×97, and ResNetDec with three macro-blocks, where it is 391×391), with pixel values ranging between 0 and 1. To enhance clarity, a colormap is employed; specifically, the "JET" colormap from OpenCV is utilized (see [27]). Consequently, pixels with higher probabilities of belonging to a particular region tend towards red, while lower probabilities tend towards blue.

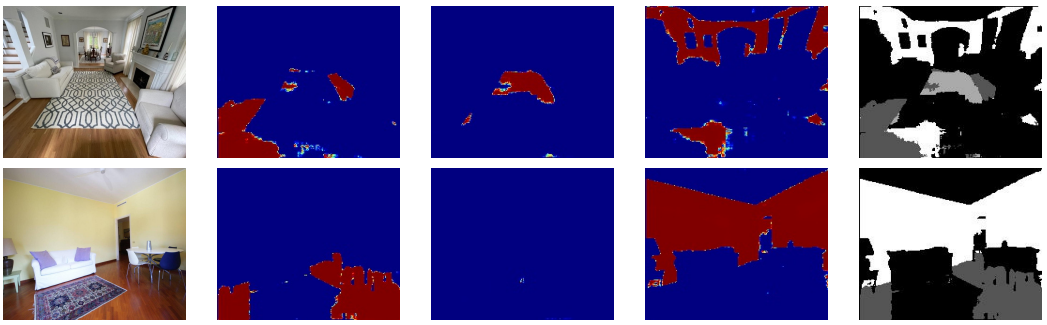
TABLE 4.3: ResNetDec only transpose convolution results.



Upon observing the qualitative results of the initial baseline, which exclusively employs two transpose convolutional layers, a significant margin for improvement becomes apparent despite the good results obtained, which indicate the goodness of the direction taken. The heatmaps pertaining to the wall region exhibit ambiguity and lack well-defined boundaries. Furthermore, the model demonstrates difficulty in accurately detecting or delineating the carpet region.

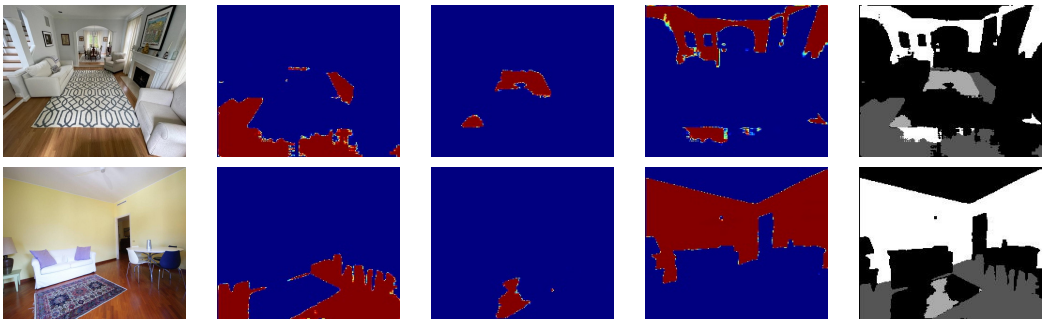
To reduce these weaknesses, an augmentation in the number of parameters is considered necessary. Consequently, a sequence of two convolutional layers is introduced subsequent to each transpose convolutional layer. This strategic modification is anticipated to enhance the model's capacity for distinguishing finer details and improving overall segmentation accuracy.

TABLE 4.4: ResNetDec without Convolutional Block results.



The analysis of Table 4.4 suggests that although the boundaries exhibit a relatively improved definition with the integration of convolutional layers, this architectural adjustment does not translate into enhanced detection of the carpet region. In addition, the edges of the regions in cluttered areas remain inaccurate. Further enhancements are needed to address this limitation within the architecture.

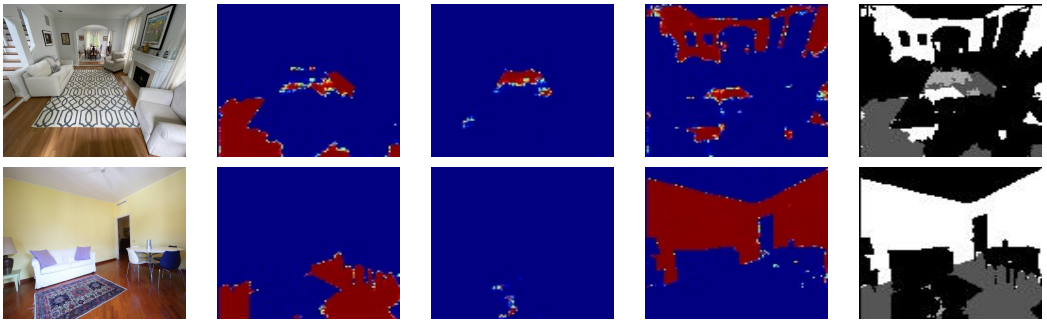
TABLE 4.5: ResNetDec 2 without skip connection results.



Referring back to the results delineated in Table 4.5, which derive from the architecture of ResNetDec incorporating both Convolutional Block and Transpose Block, it becomes apparent that this configuration achieves a notable refinement. Specifically, this architecture

reaches the minimal requisite number of parameters necessary for accurately detecting the carpet region and delineating its boundaries with greater precision.

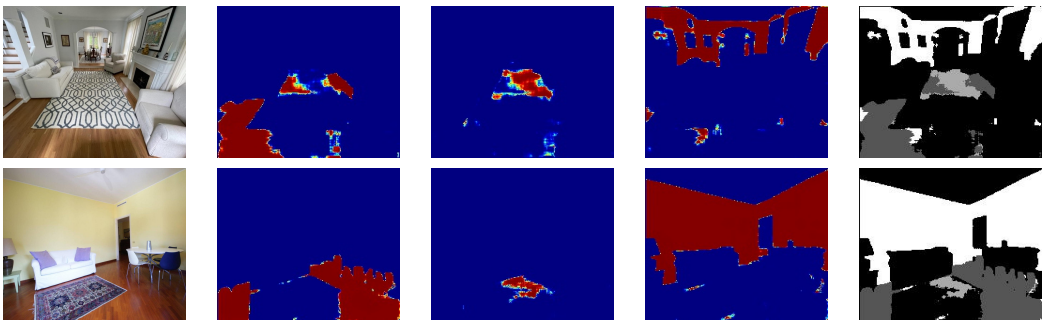
TABLE 4.6: ResNetDec with 1 block results.



Upon examining the results presented in Table 4.6, it is evident that the introduction of skip connections in ResNetDec contributes to notable enhancements, particularly when juxtaposed with ResNetDec without Convolutional Block (refer to Table 4.4), which is the architecture closest in terms of parameter count to ResNetDec with one macro-block. These improvements persist even when considering the quantified metrics in Table 4.1.

However, despite these advancements, two primary challenges persist within this architecture. Firstly, it still struggles to effectively detect the carpet region. Secondly, the heatmap it generates is one-quarter larger in resolution compared to its closest competitor, resulting in an expanded receptive field relative to the original image. This enlargement may consequently lead to increased approximation errors, thereby necessitating further refinement.

TABLE 4.7: ResNetDec with 2 block results.

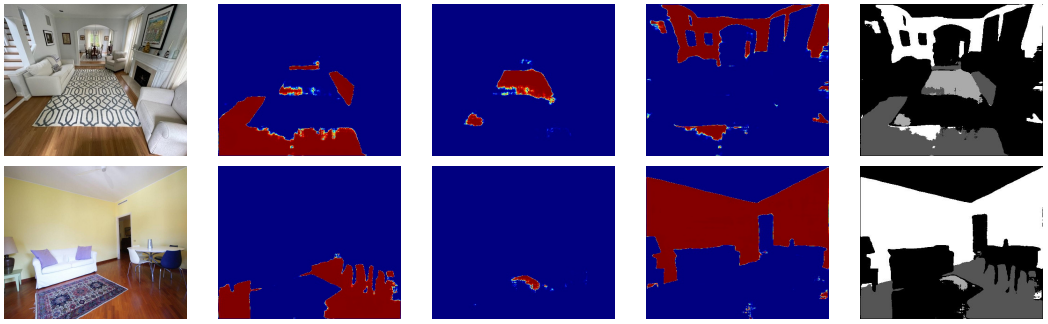


Upon investigating the outcomes obtained from the proposed ResNetDec, as delineated in Table 4.7, it becomes apparent that the architecture has attained satisfactory performance

levels, aligning with the intended objectives for which ResNetDec was conceived and developed.

To ascertain whether further augmentation of ResNetDec yields substantial improvements, as indicated by the quantified metrics in Table 4.1, it is prudent to examine the results depicted in Table 4.8.

TABLE 4.8: ResNetDec with 3 block results.



Finally, comparing images extracted from Table 4.8 with those from Table 4.7, it is observable that the two models exhibit a high degree of similarity in their performance, despite the former exploits a significantly higher parameter count. This observation prompts the need to justify the incorporation of skip connections in ResNetDec macro-blocks.

While it may seem that skip connections offer marginal benefits, particularly when contrasting the results from Table 4.7 with those from Table 4.4, further examination reveals their significance. Tables 4.2 and 4.1 demonstrate that skip connections contribute to the improvement of both mean Intersection over Union (mIoU) and accuracy metrics for ResNetDec. Moreover, their impact on computational performance is minimal, as evidenced by the negligible increase in memory usage and the insignificant difference in inference time.

Therefore, despite the apparent minimal differences in qualitative performance, the inclusion of skip connections in ResNetDec macro-blocks is justified by their tangible contributions to enhancing the model's accuracy metrics without significantly compromising computational efficiency, as states in Table 4.2.

4.3 Sampling points over heatmap

Once the method for calculating heatmaps has been developed, it remains essential to find a method that samples points from it, taking into account that a few points can be ambiguous while too many points can put segment-anything in trouble. This section will eviscerate the basic aspects of the methods considered to sample points for SAM effectively.

4.3.1 Comparison

The peculiarity of the sampling task is that the results can be evaluated only after feeding segment-anything with the sampled points to compute the mask of the region. Every presented approach will be evaluated in the SAFLOW's Sampling ResNetDec presented in Section 4.4.3.3.

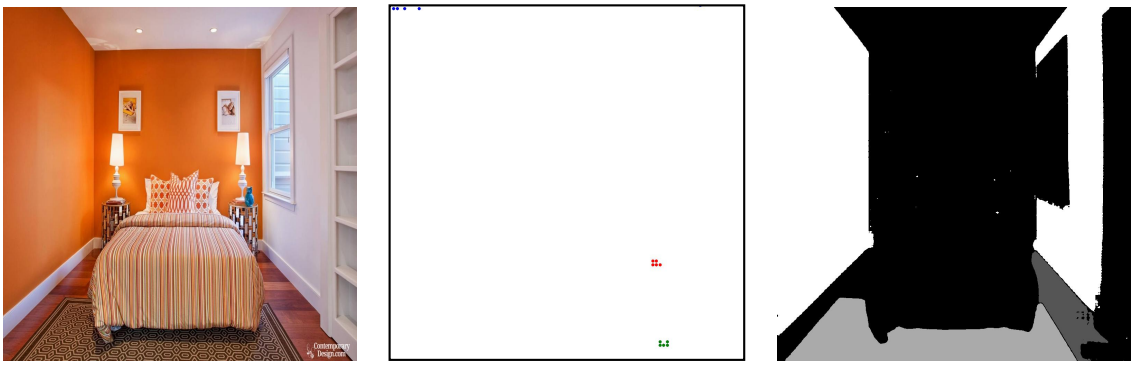
4.3.2 Top-K

The first presented approach, and the most intuitive one, involves sampling k points from each region of interest based on their probability of belonging to a specific category. This method capitalizes on the heatmap's confidence in classifying these points, leveraging the probability provided by models like ResNetDec. By prioritizing these points, we avoid making assumptions about their distribution within the region. However, while the network may confidently assign probabilities to these selected points, it's important to note that they may not necessarily be the optimal fit for SAM.

From table 4.9, is possible to understand the reasons why this simple approach can't be used. The mask is generated by sampling five points for each region and given to SAM's PE as input, with respect to the fact that SAM is trained with a maximum of six points in input and to avoid over-concentration of points, which will lead to a very low-quality mask.

The generated mask has good boundaries and no artifacts but misses some portion of the region (the wall in front, the floor on the left).

TABLE 4.9: Mask generated from top-k sampling approach.



The first try to avoid this behavior was to sample a bigger set of points (35 per region) and computing the convex hull on that set.

4.3.3 ConvexHull with Top 1

A convex hull is a fundamental concept in geometry and computational geometry. It refers to the smallest convex polygon that encloses a given set of points in a multidimensional space.

Mathematically, a convex hull is defined as the intersection of all convex sets containing the given points. It can be visualized as the outer boundary or envelope of the points.

Due to the point limit before the performance degradation of SAM, this approach aims to compute the convex hull of a larger set of points, composed of the most probable points for the region, and get the polygon that includes they, which means a lower number of points in total. Since we can force the polygon's points to be points of the set, we have the certainty that these points belong to the original set, i.e., the region of interest.

The set of points over which the convex hull is computed is composed of the m points with higher probability.

To compute the convex hull of a given set of points, we relied on the powerful Python library called SciPy [28]. Known for its extensive use in scientific computing, SciPy leverages the sophisticated algorithms presented in Qhull [29] to accurately compute the convex hull.

In taking into account the sampling of points based on the convex hull, a question that cannot be unconsidered is the probability of points. Another approach to sampling ResNetDec is to take as a set of points always the m points with higher probability, but in this case including also the point of the region with the higher probability (the most probable of all). Taking into account the normal distribution, it is probable that in the simple convex hull approach, the most probable points will be excluded, while instead could be important information for SAM.

TABLE 4.10: Mask generated from convex hull and top k approach.

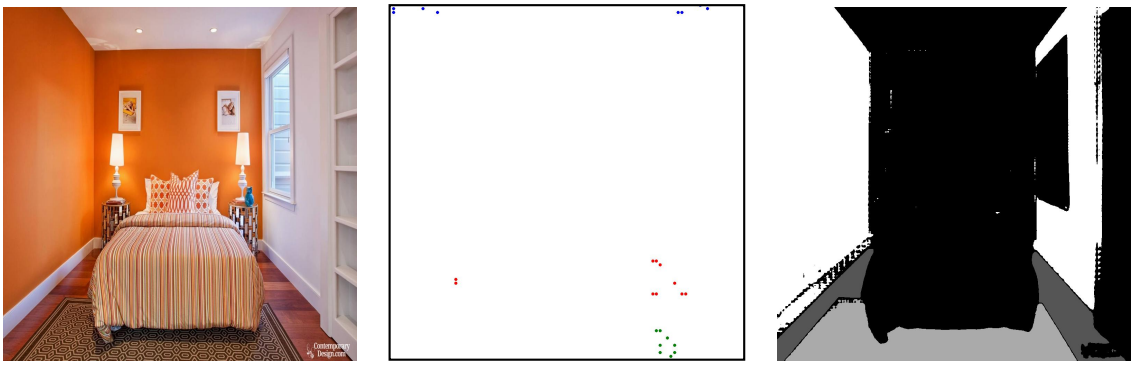


Table 4.10 evidence the fact that not all the portions of a region will be taken into consideration in the resulted mask, while feeding segment-anything with a set too large of points degrades the quality of the generated mask a lot, like explained in Section 2.2.2.2.

4.3.4 K-means

To conclude the analysis of this approach, further discussion of how hyperparameters are chosen is necessary. Specifically, the hyperparameters that impact the behavior of this algorithm are the total number of clusters and how many points are taken into account to compute the centroids based on how many of those are assigned to that region.

This process has been done empirically, trying different combinations of the number of centroids and the number of points per centroid (PPC). During the examination, stood out that a lower number of centroids with a low number of PPC leads to minimal variability, bringing the points, hence centroids, too close together to best capture the distribution of regions in

the scene. Increasing PPC while maintaining a fixed and low number of centroids partially solves the problem. To conclude, also a little increasing of the number of centroids is necessary to fully cover the region. The optimal configuration found evaluating qualitatively the results is a total of 20 centroids (proportionally divided by regions) and 50 PPC, Recall from Section 3.2.1, that each centroid will be associated with a clustroid. SAM will segment the region by composing the set of points from the union of all clustroids.

4.4 SAFLOW: Floors&Walls segmentation approaches

This section will present every approach used for automatically segmenting floors and walls, each one of those starting from segment-anything. The set of approaches will be presented in chronological order, mirroring the sequence in which they were explored. During the exploration, we need to enlighten the main constraint imposed by the company: the only input available is the image furnished by the user, nothing else.

In the course of this section, the starting baseline, the construction, in temporal order, of the evaluated approaches will be presented, and only after the end of the presentation of the approaches will they be evaluated and analyzed from both qualitative and quantitative perspectives.

4.4.1 Training details

Although different approaches are presented, the trainable components - in the cases where there are any - are all trained with a composition of Binary Cross-Entropy, which is the main component, summed with Boundary Loss (presented in [30]), which weighs the error based on how far the error is from the edge. Boundary loss is necessary for artifact prevention (see section 4.4.3.2).

$$L_{comp} = \lambda_b BCE(x, y_{mask}) + \lambda_{bdloss} BDLoss(img, y_{distmap})$$

where

- BCE is the Binary Cross-Entropy
- $BDLoss$ denotes the Boundary Loss [30].
- x the generated segmentation mask.
- y_{mask} the ground-truth segmentation mask.
- $y_{distmap}$ the distance map [30].

For training the components, in this work, λ_b equals 1 and λ_{bdloss} equals 0.1. The learning rate is 0.0001 and the optimizer used is $Adam$ with β_1 equals to 0.9 and β_2 equals to 0.999. For each approach, the training is divided in two different training. The first training is performed on ADE20k which warms up the weights of the networks, and the second training is on the Maticad indoor dataset. The reason for the division is already explained in Section 4.2.3.

4.4.2 Baseline

The Figure 4.2 is the minimal representation of SAM, that recalls its scheme in Figure 2.10, since the focus of the presentation of this set of baseline isn't concerned with what is outside the blocks instead of other details, for clearness is necessary to present a clean pipeline with only the segment-anything's main component. The segment-anything main components are the image encoder, which given an image computes its embedding - the activity that requires the most time and resources -, the prompt encoder, which codifies the input provided externally in addition to the image, and the mask decoder that outputs the binary mask(s) and its(their) predicted mIoU [Section 2.2.2.2]. As the initial baseline the native version of segment-anything depicted in Figure 4.2 isn't feasible due to its double input nature, as previously explained in Section 2.2.2.2. So, in the continuation, the evolution of the approaches explored for adapting SAM is presented.

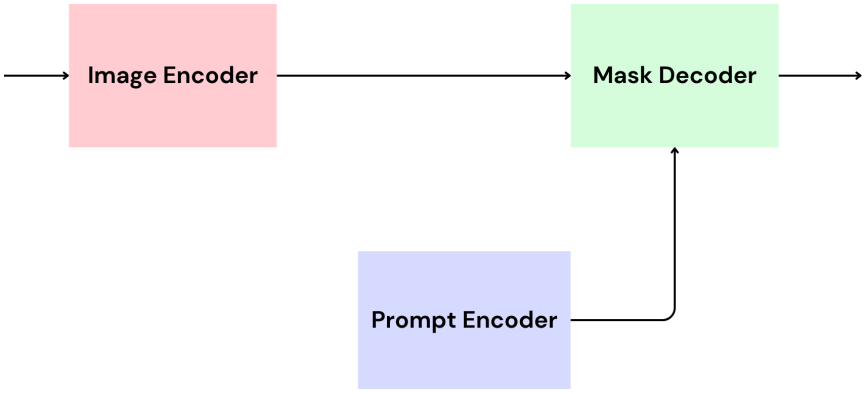


FIGURE 4.2: Baseline Architecture.

In each approach, the input is an RGB image and the output is a binary mask that segments a single region. This means that each approach manages the segmentation of multiple regions differently from the others.

4.4.3 Single component approaches

This set of approaches demonstrates the efficacy of leveraging a singular component to proficiently accomplish the desired objective.

4.4.3.1 Learnable parameters

The learnable parameters approach involves furnishing the mask decoder with a trainable tensor, serving as a guide for segmenting specific regions rather than relying solely on prompt embeddings. This tensor is designed to encapsulate the segmentation area independently on the input image, ensuring its adaptability post-training. Essentially, it furnishes a semantic blueprint for the segmentation target. They have the abstract and general meaning of the region for which the mask decoder has to furnish a segmentation mask.

In this manner, this approach doesn't require any additional input, so, as shown in 4.3, those learnable parameters replace the prompt encoder, which within this approach, is useless.

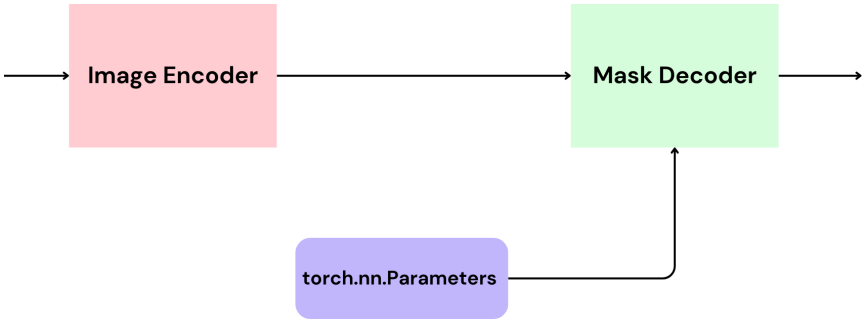


FIGURE 4.3: Learnable parameters Architecture.

4.4.3.2 Leveraging encoder space

Evolving from the previous approach, in which the learnable parameters that substitute the original prompt encoder are image-agnostic, it is natural to explore a way to make the substitution of the prompt encoder image-aware.

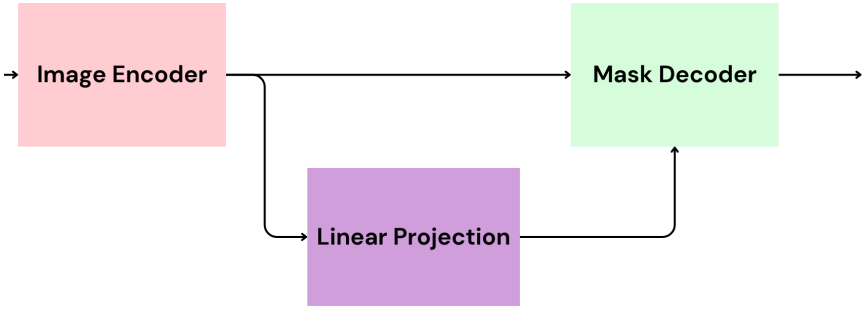


FIGURE 4.4: Linear projection Architecture.

To achieve image-awareness, it's crucial to leverage image-specific techniques. Within this framework, is essential to deal with the SAM's Image Encoder space. To accomplish this, we utilize the embedding representation of an image as input for SAM (Segment-Anything Model) and subsequently linearly project it into a new space. Semantically, this space is expected to be closely aligned with the prompt encoder's space, although this alignment process isn't explicitly facilitated.

Given that the image's embedding assumes a shape of $[B, 256, 64, 64]$, where B denotes the batch dimension, our initial step involves flattening the last two dimensions to yield a vector of shape $[256, 4096]$. Subsequently, we perform two distinct and independent linear

projections (one for dimension) to maximize the information retrieved from the original embedding. This approach enhances the interpretability and efficacy of our image-awareness strategy.

Artifacts However, this approach generates artifacts in the segmentation masks, reducing the quality of the result. Removing the prompt encoder, no matter how small it is, means removing a component trained over billion of input per epoch, and consequently, the mask decoder expects a prompt embedding with a precision that we can't achieve due to the size of our annotated dataset, which is almost $6,000$ times smaller [Section 4.1.3]. Another important factor to consider is that if SAM is tested with simple points as inputs, it doesn't generate any artifact, and the resulting mask is of high quality, meaning that SAM is capable of doing what is required. So the path to follow now is not to try removing artifacts acting on improving something that is trained from scratch or with post-process operations, but releasing the full power of native SAM improving the quality and the meaningfulness of the inputs furnished. A more in-depth analysis of artifacts will be made in the qualitative results section 4.4.6

4.4.3.3 Sampling from ResnetDec

Upon recognizing the indispensable nature of each component within the segment-anything model for maintaining optimal quality performance, a paradigm shift becomes necessary. Preserving the original structure of segment-anything entails either altering the user experience of Maticad's current pipeline or developing an adapter to emulate user actions and automate them.

It is at this juncture that ResNetDec [3.1] is introduced as an integral component within the pipeline, rather than as an entity isolated from others.

The most straightforward approach involves harnessing ResNetDec in conjunction with SAM's image encoder. ResNetDec yields a set of points delineating a region, which are subsequently encoded by the prompt encoder. Alongside the input image embedding, these encoded points

are fed into the mask decoder to produce the region's mask. Figure 4.5 presents this approach's architecture.

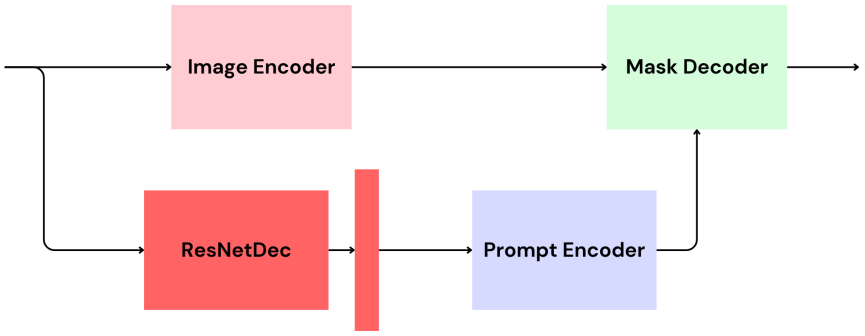


FIGURE 4.5: SAFLOW w/ ResNetDec Architecture.

ResNetDec, coupled with its sampling techniques, introduces a set of novel approaches. Instead of attempting minor adjustments, exploiting it fully enables the capabilities of SAM, aiming to enhance its functionality. Even in this basic configuration, leveraging ResNetDec significantly enhances the quality of masks while drastically reducing the occurrence of artifacts. The approaches that exploit ResNetDec have the possibility of examining the heatmap to detect the extension of a region, in order to filter out the excessively small ones. Also other analysis can be done on top of that, like in the sampling case [Section 3.2].

4.4.4 Multiple components approaches

Relying solely on points sampled from ResNetDec effectively diminishes the occurrence of artifacts but may sacrifice some crucial high-level image information. A concentration of points overly centered in a specific area or affected by approximation errors can result in a segmentation mask with altered semantic interpretation. With this understanding, it becomes apparent that leveraging multiple components for the SAM's mask decoder has the potential to generate a more accurate segmentation mask.

4.4.4.1 Align encoder space with points embedding space

In this approach, a linear projection similar to the one described in Section 4.4.3.2 is trained, this time referencing also the prompt encoder. The SAM's image encoder space is still utilized to extract high-level information from the input image. However, the goal here is to move this information closer to the encoding space of the prompt encoder. This adjustment takes place during the training phase. During inference, the prompt encoder component is omitted, relying on the assumption that the intermediate linear projection has effectively learned to align image features more closely with it.

Given that the output of the prompt encoder manifest an image-aware nature, ResNetDec was once again employed. In the training phase and while computing image embeddings, we simultaneously compute the heatmap of the input image, sample points based on the region of interest (as detailed in Section 3.2), and compute their embeddings. These prompt embeddings then serve as a reference for the output of the intermediate linear projection.

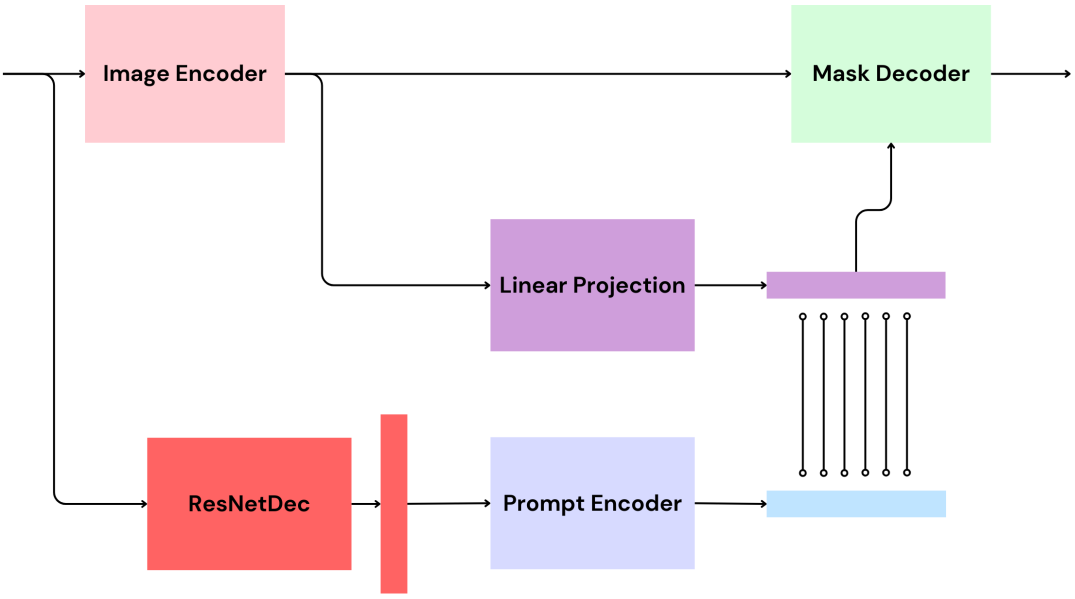
This methodology draws inspiration from the knowledge distillation concept introduced by Hinton et al. [21].

At a conceptual level, the objective is to instruct the linear projection to extract high-level features representing the region of interest and locate them closer to the embeddings of points within that region. By doing so, we provide a reference for how the output should align with the SAM's mask decoder, mitigating artifacts as discussed in Section 4.4.3.2.

In Figure 4.6 shows the scheme of this approach, enlightening that both the prompt encoder and ResNetDec are components of support during training and not exploited at inference time.

Alignment Training details Due to the distillation component, for this approach, the training details are slightly different. The training process is still divided on ADE20k before and Maticad indoor then, but the loss function must take into account also the distillation

FIGURE 4.6: Align the intermediate linear projection to the prompt encoder architecture.



ResNetDec and Prompt Encoder will be used only during training phase.

component. Given that the output binary mask is defined as follows:

$$mask = MD(LIN(IE(img)), IE(img))$$

where

- MD is the SAM's Mask Decoder
- img is the input image
- MD, LIN and IE are respectively the Mask Decoder, the linear projection and the Image Encoder

The loss used for training is the following one:

$$L = \lambda_{kd} MSE(x, PE(points)) + \lambda_c(\lambda_b BCE(mask, y_{mask}) + \lambda_{bdloss} BDLoss(mask, y_{distmap}))$$

where

- BCE is the Binary Cross-Entropy
- $BDLoss$ denotes the Boundary Loss [30].
- MSE is the Mean Square Error.
- x the linear projection output.
- y_{mask} the ground-truth segmentation mask.
- $y_{distmap}$ the distance map [30]
- PE is the SAM's prompt encoder
- $points$ are the set of points sampled from ResNetDec

The value of λ_{kd} is 0.1, the value of λ_c is 0.3 and the values of λ_b and λ_{bdloss} are the same of the original training configuration, respectively 1 and 0.1. Is still used the $Adam$ optimizer with the same values as before.

4.4.4.2 Iterative Mask Refinement

This approach is the proposed method for SAFLOW and is presented in Section 3.3.

4.4.4.3 Iterative Mask Refinement with Erosion

Building upon the Iterative Mask Refinement approach, this one delves deeper into comprehending the influence of the initially generated mask on the subsequent iterations. Its primary objective remains the reduction of artifacts produced by SAM, both in terms of quantity and extent.

This strategy endeavors to erode the first binary mask generated of each region, thereby diminishing the likelihood of boundary errors and enhancing the coherence and compactness of the segmented regions. It is hypothesized that this erosion process could yield improved performance outcomes.

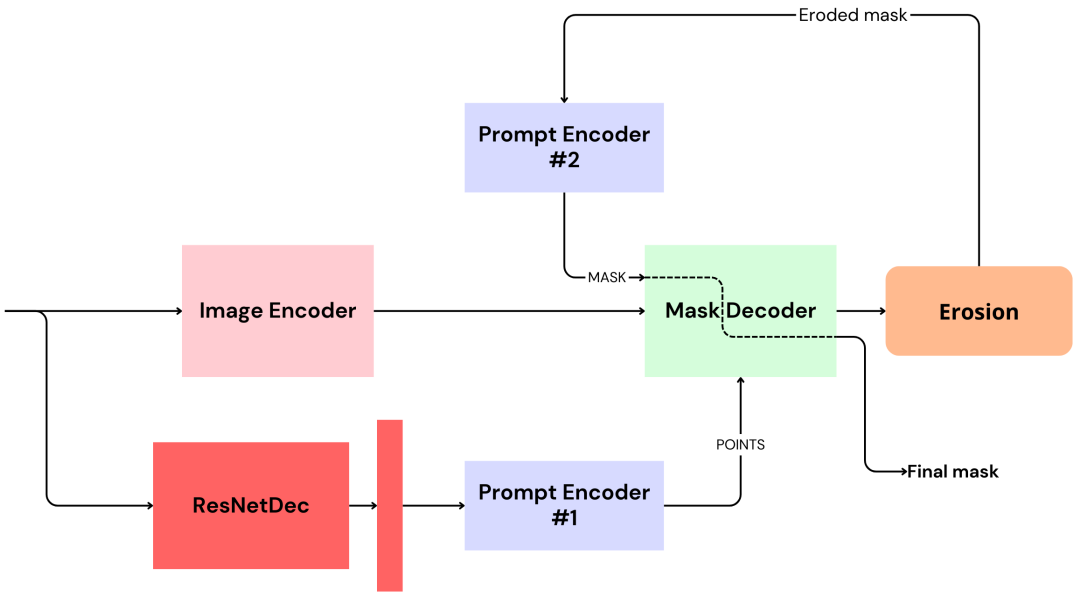


FIGURE 4.7: Iterative Mask Refinement with Erosion approach.

The erosion operation in image processing can be conceptualized as a form of 2D convolution operation. Within this operation, the resulting value at each pixel in the output image is determined by applying a ”minimum” operation over the corresponding window or kernel in the input image.

It serves as a morphological filtering technique used for tasks such as noise reduction, edge detection, and object extraction. By performing a minimum operation, it effectively removes small-scale features while preserving larger structural elements within the image. This characteristic makes it particularly useful in various image-processing applications where the emphasis lies on delineating object boundaries or extracting salient features. For this reason, this approach is born.

4.4.5 Quantitative results

In this section, a quantitative comparison between the presented approaches is performed. In Table 4.11 each method will be evaluated by computing the Mean Intersection Over Union (mIoU) and the accuracy over two different datasets, while in the continuation a qualitative comparison and analysis is done.

TABLE 4.11: Floors&Walls segmentation Ablation study results. Approaches Comparisons.

	ADE20k		**Maticad Indoor**	
	mIoU	Accuracy	mIoU	Accuracy
Single component approaches				
Learnable parameters	0.727	0.829	0.870	0.926
Leveraging encoder space	0.695	0.802	0.876	0.929
Sampling from ResNetDec	0.741	0.834	0.866	0.925
Multiple components approaches				
Align embedding to prompt encoder space	0.729	0.828	**0.904**	**0.948**
IMR with erosion	0.746	0.838	0.874	0.929
SAFLOW (IMR)	**0.750**	**0.839**	0.888	0.938

Please note that Maticad Indoor has only three classes, while ADE20k has four, but adjusted according to the model.

From a quantitative point-of-view, the IMR approach is the best on the evaluation over the ADE20k dataset, while is slightly worse in Maticad Indoor with respect to the Alignment approach. The reasons why the proposed method is IMR and not Alignment should be sought in the qualitative analysis of the results. A foretaste: Alignment is constrained to 3-class segmentation while IMR is not, and from a qualitative point of view, IMR has far fewer artifacts. Even if the mIoU is higher, it does not directly mean that the quality, at least perceived, is better. And since the quality aspect is more important to Reality Remod, the choice after the qualitative analysis will be trivial.

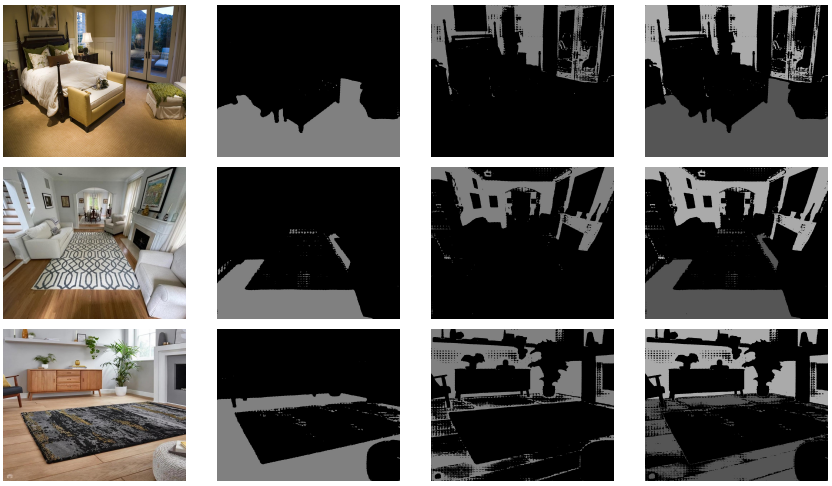
4.4.6 Qualitative results

In this section, we embark on a qualitative evaluation of different approaches, crucial for gaining insights into their usefulness. For consistency, we'll focus on three specific images, facilitating a thorough comparative analysis. The presentation order will be the same as the evolution during this thesis work, aiming to provide clarity regarding the reasons that led to the final proposed method. Every image used for qualitative comparison is taken from the Maticad Indoor dataset [Section 4.1.3].

To better understand the results, we must first specify that Maticad, at the beginning of the project, specified to us that in the Maticad Indoor dataset, floor masks also include carpet. This may seem like a detail, but in the further analysis of qualitative results, it will be an important discriminator. For this reason, in the presentation images of the results, two out of three images contain carpets.

The first approach explored is the simplest one, the **Learnable parameters** approach - explored previously in this section.

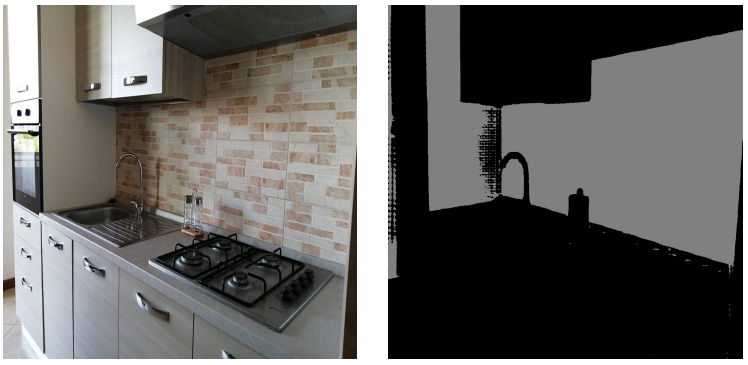
TABLE 4.12: Qualitative results of Learnable parameters approach.



As can be seen from the Table 4.12, the method can detect correctly the region, but fails resoundingly in being precise in the boundary of the regions. Another factor that underscores the poor quality of the masks generated by this approach is the heavy presence of **artifacts**.

Artifacts are SAFLOW's main problem since even a relatively small amount of them renders the mask unusable for final purposes. Some post-processing related to this approach has been attempted, but without obtaining significantly better results, so for this reason they will not be presented. In Table 4.12 the artifacts are present in every segmentation mask relative to the wall region - the third column - but for better understanding it in Figure 4.13 another example is given, where the presence of artifacts is not particularly massive, but still impactful.

TABLE 4.13: Another examples of artifact generated from the LP approach.

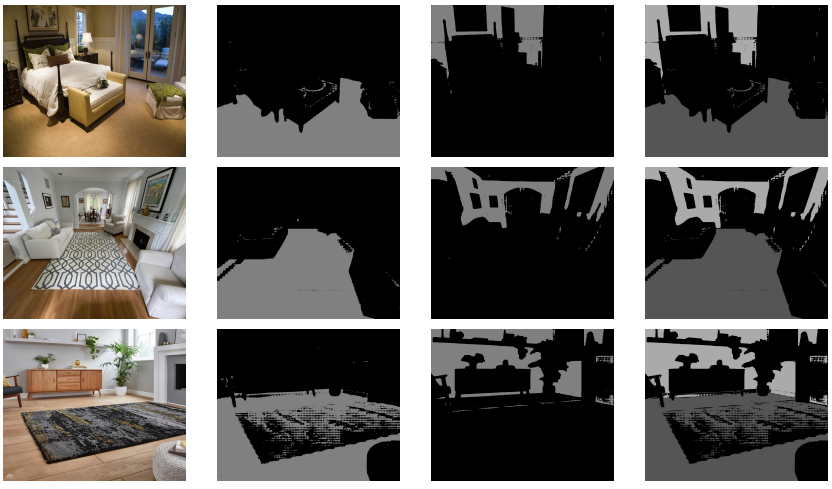


Since SAFLOW's learnable parameters approach is based on the assumption of learning a vector that represents the abstract concept of the region instead of some image-aware information, from now every input given to the SAM's Prompt Encoder or MD will be image-related.

Starting from the approach of SAFLOW's learnable parameters, it comes naturally to think of extrapolating image-related information from SAM's Image Encoder encoding. The **Linear projection** approach, presented in Section 4.4.3.2 improves the situation both at the level of precision in contouring the edges of regions and at the level of artifacts generated.

The results of SAFLOW's linear projection approach are presented in Table 4.14.

TABLE 4.14: Qualitative results of Linear projection approach.



As can be seen from Table 4.14, the situation slightly improves, and it is at this point that joining carpets and floors in the same mask becomes a problem. SAM, since it segments one object at a time - which class cannot be specified - tends, rightly, to segment individual objects or individual regions. Carpets and floors, though semantically related, are two quite distinct regions. Forcing SAM to merge these two regions requires consideration of alternative approaches.

One approach that can certainly be exploited is to augment the training set with more images related to this case study. But since SAM has been trained on a huge amount of data [Section 4.1.1], imposing this while maintaining its generalization capabilities is a particularly daunting task.

In the continuation, we will explore solutions that are not based on increasing the amount of training data.

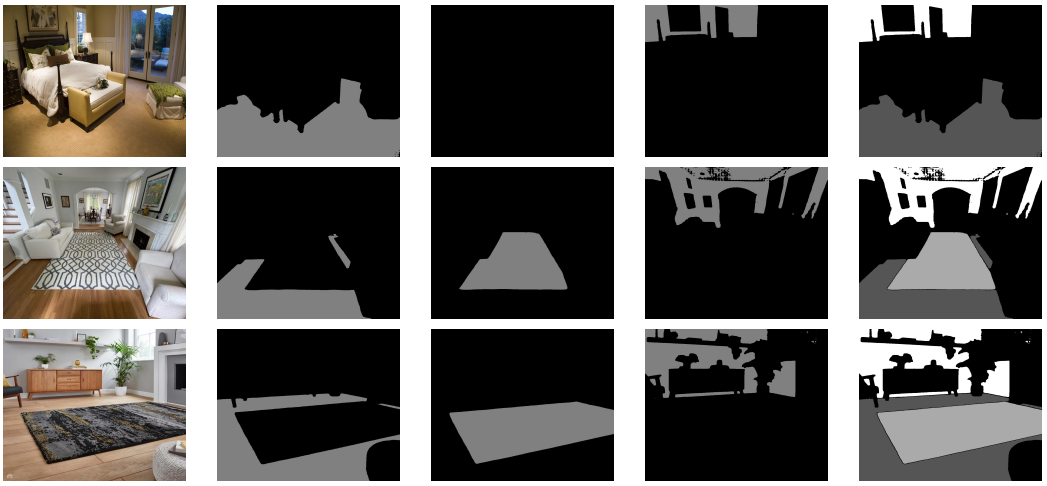
It is at this point, chronologically, that ResNetDec (see Section 3.1) was developed and introduced within its sampling method (in section 3.2). Through ResNetDec, it is possible to separate the segmentation of the two distinct classes - floors and carpets - and then give Maticad the option of merging them after the fact or not, depending on the customer's customized needs.

Also, as explained earlier, giving input to the SAM's Mask Decoder of an PE-generated embedding rather than something trained with relatively little data is something that blends better with SAM.

Going further, the **ResNetDec sampling-based** approach (see section 4.4.3.3 was analyzed in the most intuitive and simple way possible.

As can be seen from the Table 4.15, the situation has definitely improved. Now the network is able to separate the two ambiguous classes in the dataset - carpets and floors -, greatly reduce the amount of artifacts (though still present), and contour the edges with vastly superior quality. It can still be improved, which is why we have gone on to explore how to integrate ResNetDec (which has become essential at this point) with SAM.

TABLE 4.15: Qualitative results of base ResNetDec approach.



At this point, the exploitation of ResNetDec becomes mandatory.The changes to the resulting mask are too evident to consider excluding it from the final approach. The non-trivial aspect is how to integrate it to achieve the best possible results.

The next approach explored is the Alignement between the SAM's Image Encoder space and SAM's Prompt Encoder space (explained in section 4.4.4.1). Before abandoning the idea of exploiting SAM's Image Encoder coding space, one must verify that it brings no benefit. Conceptually, being able to extract high-level information from the encoding of a Vision Transformer, as has been done in the Linear Projection approach, is not something taken for granted.

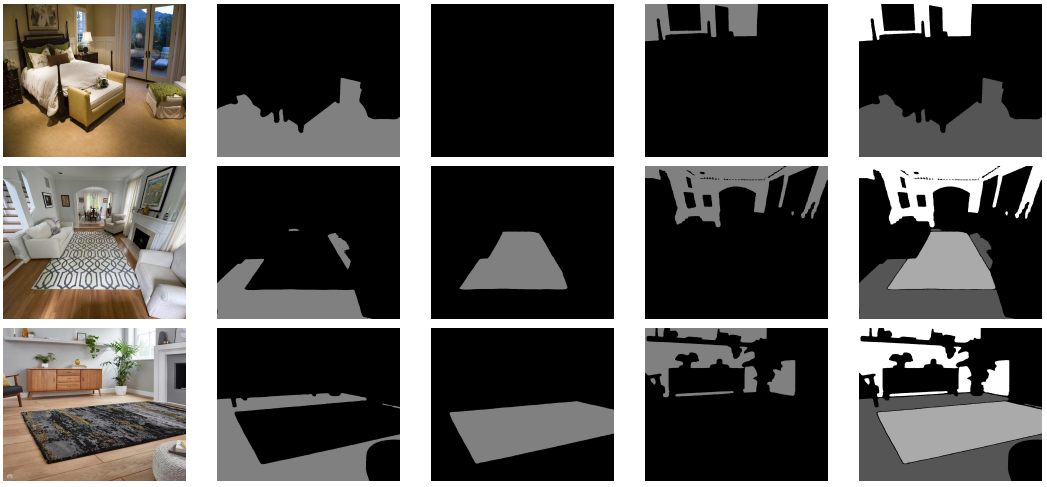
This is the main reason for the second alignment attempt presented and whose results are in Table 4.16.

As much as the results are markedly improved over its predecessor, the Linear Projection approach (see Section 4.4.3.2), we can see that the masks are more precise, better defined, and with relatively few artifacts, the inability to separate carpets and floors is a major limitation that requires us to set aside the exploitation of SAM's Image Encoder code space.

The last attempt to improve the result leveraging ResNetDec is the **Itertive Mask Refinement** approach, presented in Section 3.3.

TABLE 4.16: Qualitative results of Alignment approach.

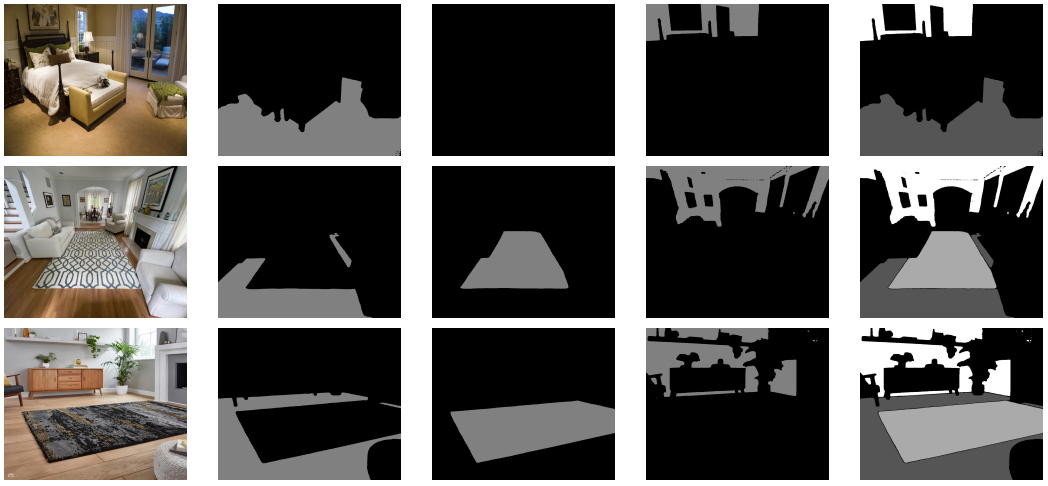
TABLE 4.17: Qualitative results of IMR approach.



The improvements made by this approach over the simpler sampling approach are more subtle. This is not to say that the situation does not improve; on the contrary. From the second image it can be seen that in truth this approach makes slight improvements on the issue of artifacts. In this approach, considered the best after qualitative and quantitative analysis of the various approaches, artifacts are still present, although to a much lesser extent. Most of the images analyzed with this approach have no artifacts of any kind, while those that do have them are usually small and located at the intersection with the wall.

The last explored approach aims to further improve masks by reducing artifacts. This approach is explained in the 4.4.4.3 section and basically adds an erosion operation on the first mask. It is called **Iterative Mask Refinement with Erosion**.

TABLE 4.18: Qualitative results of IMR with Erosion approach.



These latest results show that the situation remains essentially unchanged, demonstrating that the highest quality achievable by this type of approach has been achieved. Since the masks obtained are of a very good quality anyway, the exploration of other approaches stops at this point.

4.4.7 Overlap problem

Since segment-anything can classify one single region each time, it is mandatory to merge the masks of the various regions, obtained independently, into one representing the overall segmented image, dealing with overlapping. It is intuitive that if there is an overlap there is also an error in the generated mask since the classes that are taken in consideration are mutually exclusive - a pixel can not be both wall and floor.

Experimentally, the fact arises in those images where a region is missing, like a photo of a floor without walls or vice-versa, like the one depicted in Figure 4.19. It is not a general rule, in most of the images where a region is missing it occupies an extremely small portion of the image, SAFLOW behaves correctly.

This problem mainly arises when a heatmap generator like ResNetDec is not employed but it can still arise, even though to a much lesser extent.

As already stated above in this Chapter, ResNetDec gives the possibility to make some analysis over the region presented in the image, and if ResNetDec doesn't detect any floor or wall, the final output of SAFLOW won't be affected by the missing region in the original image, like it isn't affected from images without carpet, also if conceptually is slightly different. In the version of SAFLOW where the heatmap generator is not leveraged, this problem has a greater impact.

Due to the nature of segment-anything, that can compute a binary mask, can happen in the final mask - the one obtained after merging all individual masks - that a portion has overlapped region.

While exploiting ResNetDec creates a correlation between the regions in the image, also if at an early stage and not in the actual process of the generation of the mask, helps to reduce a lot of the overlap problem, that now will be presumably in the border, the architectures that have not a heatmap generator this mutual exclusion, however minimal it is, is absent, this leads this class of architecture to suffers in this scenarios. It is clear from the results in the Table 4.19 how differently this problem is handled in the two approach classes.

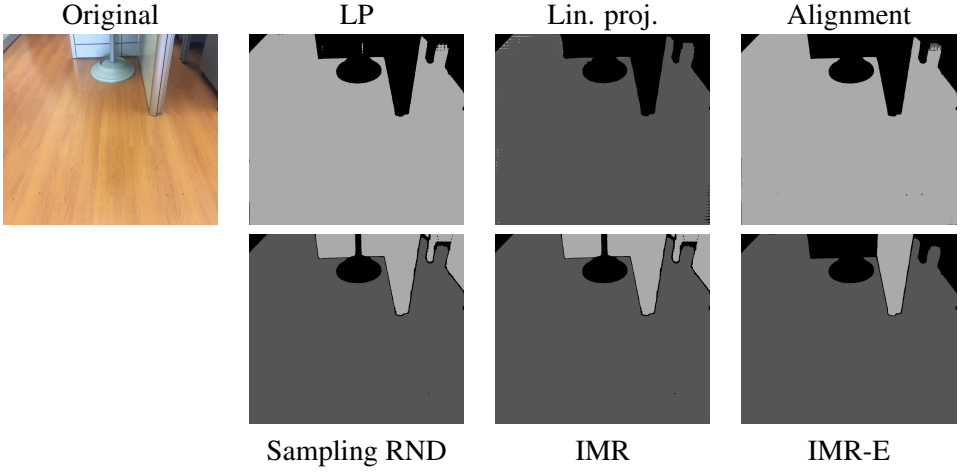


TABLE 4.19: Example of the Overlap problem.

Up to now, in the final pipeline presented, the handling of the overlap problem is based purely on the probability of the pixel belonging to one class rather than that other, as is normally done in segmentation. The problem is that usually this process is performed under assumptions that are different from those of SAFLOW. There is no correlation whatsoever between two SAM-generated masks at two different times, even though they theoretically belong to two different objects, this can lead this overlap management process to go wrong.

There is no reason why the probability of the same pixel but in two different masks - before thresholding - are correlated. So a pixel may have a higher probability for one even if it is wrong because the probabilities coming out of SAM do not have the classical assumptions of segmentation.

This method that deals with the overlap problem does not even take into account the concept of spatial coherence, that is essential for this purposes.

4.5 InPainting models: a comparisons

At this point, a model of inpainting needs to be chosen, and the candidates are MIGAN and lama, presented respectively in section 2.3.3.2 and 2.3.3.1. The comparison will be done both in a qualitative and quantitative way. The quantitative study will involve the analysis of both performances and the perceptual metrics LPIPS [19] and FID [31]. For the qualitative analysis, are taken into consideration some images from the Maticad Scene dataset, are the inpainted images generated by both models and then compared with the ground truth, recalling the fact that the images in Maticad Scene are fully synthetic.

4.5.1 Quantitative results

Table 4.20 provides a comprehensive comparison between the two prominent inpainting models under consideration. The optimal model selected from this comparison will be integrated into the final proposed pipeline.

TABLE 4.20: Inpainting networks comparisons.

The values in 4.20 relatives to the Places2 dataset are taken from the works of Sargsyan *et al.* [15].

	Performance		**Places2***		**Maticad Scene**	
	#Params	Speed (s)	LPIPS ↓	FID ↓	LPIPS ↓	FID ↓
Inpainting networks						
MiGAN	**5.9M**	**0.226**	0.394	**11.38**	0.036	**0.0008**
lama	27M	0.498	**0.378**	22.0	**0.031**	0.0026

Analyzing the Table 4.20 is possible to observe that MIGAN has better performances, but taking a look at the FID and LPIPS metrics on both datasets, Maticad Scene and Places2, the qualitative performances need to be further investigated.

4.5.2 Qualitative results

The investigation of lama and MIGAN needs to pass through an in-depth evaluation of images. In Table 4.21 are taken three different scenes, with a binary mask delineating the object's boundary to remove, the first image reconstructed is obtained from MIGAN, the second from lama. Since the Maticad Scene dataset is fully synthetic, it is possible to compare the result of the two different models to what should be, i.e. the image without the synthetically constructed object.

Maticad Scene is considered, from Maticad, the dataset that better represents the data distribution of the application field of this work, also if the data are fully synthetic. This is the only way in which Maticad can provide us with a labeled dataset for that task. A further comparison of the model on real images will be done in collaboration with the company.

Table 4.21 underscores the superior performance of lama within the given data distribution. In the inpainted images, lama exhibits fewer artifacts compared to MIGAN and is perceived to be visually superior by human observers.

Based on these observations, lama has been selected as the preferred inpainting model to be integrated into the final proposed pipeline.

TABLE 4.21: Inpainting qualitative results.

Original Mask MIGAN lama GT

TABLE 4.22: Example of the shadow problem.

Original Mask MIGAN lama GT

The primary limitation encountered by both inpainting approaches pertains to the treatment of shadows. When the mask includes the object's shadow, the reconstructed image tends to exhibit satisfactory results. However, if the shadow is excluded from the mask, the inpainting model fails to reconstitute that portion of the image, as it lies outside the mask boundaries, leading to a degradation in the quality of the generated image. This is because since the shadow is outside the mask, the inpainting models can't remove that region anyway, so to reconstructing the image to maintain coherence with what is outside the mask, the result will be the hole filled with some darker textures instead of something more perceptually similar. This happens only when the shadow of the object is outside the dilated mask.

This challenge is particularly significant because the mask provided as input to the inpainting model is computed from SAM, which does not incorporate the shadows of the object in its segmentation process, even when explicitly addressed to do so.

Addressing this issue is crucial to enhancing the overall performance and fidelity of the inpainting process within the proposed pipeline.

4.6 SCA-based classification methods

In this section will be discussed the experiments done for the alignment of the SAM and CLIP space, as explained in Section 3.4.

4.6.1 Dataset preparation

The dataset used for precomputing the features vector is SA1B (see Section 4.1.1), since is the largest and most diverse dataset, composed by an overall of 11 million of images. However, also if the alignment of the two spaces may benefit from this high dimensionality of the dataset, it will make both training and storage management particularly complicated and time-consuming. A small version of SA1B, of about 2% of the total amount of original images, was used. Each image was resized to 1024×1024.

To limit the memory requirements during training - so as to maximize the batch dimension - the features vector of both SAM's Image Encoder and CLLIP Visual Encoder are precomputed and saved. So, instead of elaborating each time the image embedding for both models, a pre-computed version of it is used.

4.6.2 Training details

The loss used for training the Neck is the Contrastive Loss. What this loss does is to move closer, in the SAM encoding space, the embeddings of images that are close also in the CLIP space. The neck output vector is a length of 768, the same as CLIP. In detail, the first operation applied to both vectors is the L2-normalization. Then the similarity between the embeddings in the batch is computed. The neck output vector has shape $[bs, 768]$, where bs is the batch size, for computing the similarity is performed a matrix multiplication between the

neck embeddings and its transpose, leading to a batch similarity matrix of $[bs, bs]$, where in every location is specified the similarity measure between the two batch elements. The same process is done for CLIP embeddings, both matrices are multiplied by a learnable scalar, called logit scale. At this point, where there are two matrices of size $bs \times bs$. What needs to be done is to move the matrix relative to the neck towards the matrix computed from CLIP embeddings, and to achieve that, the Cross-Entropy loss is exploited, forcing the embeddings close to each other to have a closer representation, to reduce the similarity discrepancy. This process is depicted in Figure 4.8.

In conclusion, when the Contrastive Loss is becoming less powerful, it is substituted with MSE Loss, to increase the alignment precision.

To train the neck, the optimizer used is $Adam$ with β_1 equals to 0.9 and β_2 equals to 0.999, with a learning rate equal to 0.0001, a dropout of 0.3 and a batch size of 512. The d_{model} of the neck is 768. The starting value of the logit scale is $\log(\frac{1}{0.07})$. Since the neck is trained from scratch, a learning rate warmup for the first 30 iteration is performed, linearly bringing the learning rate from 0 to the value defined above. The first experiments are performed over a single NVIDIA GPU A40, with a memory of 48 GB.

As good and indicative as the first tests were, no significant results could be obtained to present at the moment, as this method requires a particularly large amount of training resources.

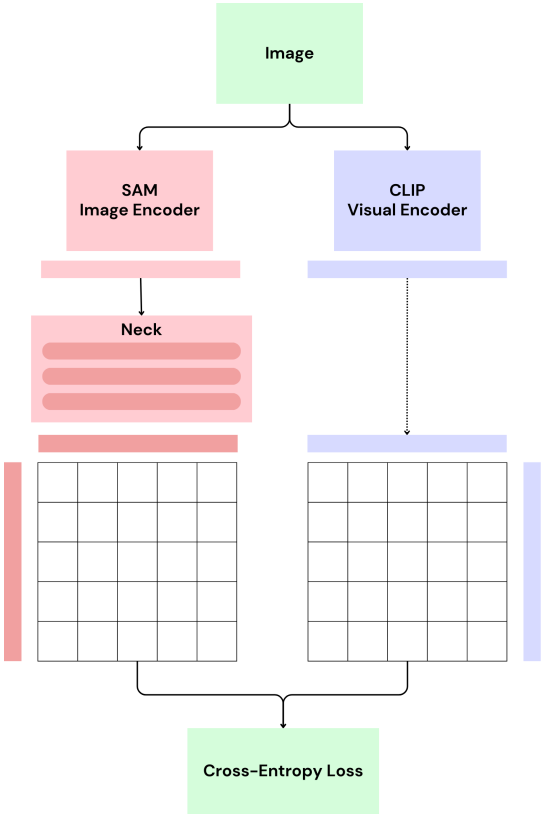


FIGURE 4.8: SCA training scheme.

Chapter 5

Results and Proposed Pipeline

In this chapter, all the various elements and components that have been discussed and presented throughout this thesis will be seamlessly integrated into a single, unified Computer Vision pipeline.

At the beginning of this thesis, several primary objectives have been clearly outlined. The first goal was to improve upon the qualitative performance of the existing segmentation network that Maticad currently has in production. This existing network, as elaborated in detail in Section 2.2.2.1, is known as the Dense Prediction Transformer. Therefore, a meticulous comparison between the proposed SAFLOW and the existing DPT will be undertaken.

The second objective, which is of equal importance, was to be able to segment any given object within an image and remove it. This removal is crucial to recalculate the segmentation mask of the floor and the walls, without the removed object. This step allows for a better appreciation of the new coverings. Additionally, this process must enables also the placement of a new 3D model of the removed object in its exact position, thereby enhancing the realism and accuracy of the image. Naturally, to understand which object has been removed, it must be previously classified. The classification component is essential for both limiting the usage to the final client of Maticad's application - avoiding the illegitimate use of the inpainting network - and for understanding which class of 3D object to allow for insertion into the scene. To achieve these multifaceted goals, the pipeline proposed is presented in detail,

outlining the steps and processes involved. For this second goal, is reported a single visual example to visually understand it.

5.1 Floors&Walls segmentation results

In this section, the quality of the proposed component SAFLOW will be assessed. To achieve that a comparison with DPT needs to be done. The comparison will take into account both qualitative and quantitative metrics, as has been done for every other comparison within this thesis.

5.1.1 Comparison with DPT

In Table 5.1 is shown the comparison. As the results highlight, DPT still comes out on top on the various metrics.

TABLE 5.1: DPT vs SAFLOW.

	ADE20k		**Maticad Indoor**		**Performance**	
	mIoU	Accuracy	mIoU	Accuracy	# Params.	Speed [s]
DPT	**0.824**	**0.898**	**0.929**	**0.963**	**106M**	**0.225**
SAFLOW	0.750	0.839	0.888	0.938	641M	2.04 (0.255)

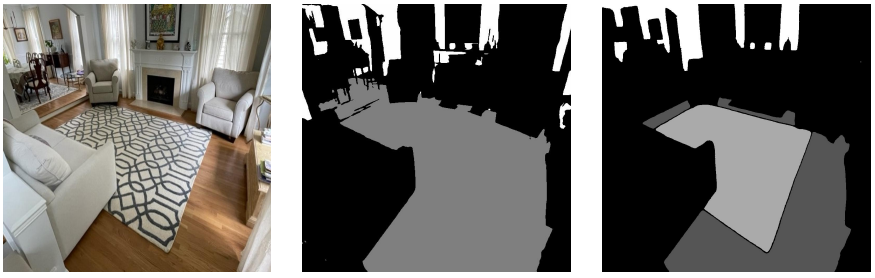
.

However, some values of Table 5.1 are misleading. First of all, DPT only segments floors and walls, while SAFLOW is an adapter for segment-anything, which as the name implies, segments anything.

This not only gives the possibility of being able to accomplish the second objective of this work but also has implications for the evaluation of the model itself on the first task. As is evident from the images depicted in the Tables in Section 4.17, SAFLOW, by separating carpets from floors, creates a separation boundary that is not handled in the presented approaches. This issue has two major implications.The first implication highlights the fact that

with SAFLOW is possible to segment carpets and floors into two different classes automatically, which DPT cannot do. This, for Maticad, as was specified during the course of the work, is very useful.

TABLE 5.2: Visual comparison between SAFLOW and DPT.



Original DPT SAFLOW

The second is that in the ground truth of the two datasets used for comparison - Maticad Indoor 4.1.3 and ADE20k 4.1.2 - this margin of separation is not particularly marked, if present (e.g. in Maticad Indoor the carpet mask is fused with the floor one). This factor inevitably involves worsening the qualitative metrics, but it does not mean that the masks are necessarily of lower quality, at least from a certain point of view.

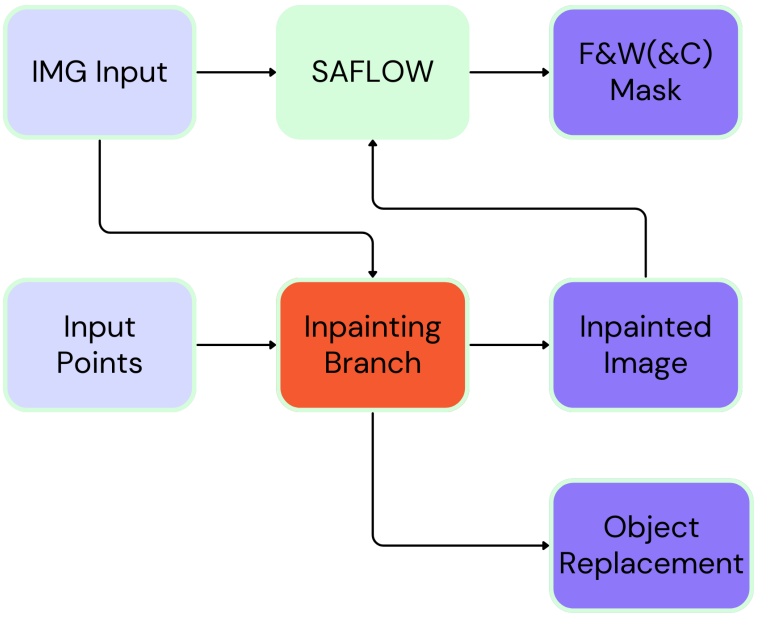
In Figure 5.2 are shown the differences between the two generated masks.

A similar discussion can be made in evaluating the number of parameters used by the two models. SAM uses 641 million while DPT 106 million. This difference is justified by the two extremely different capabilities, given the different nature of the two models. DPT with 106 million parameters segments only 2 classes, while SAM with 641 million segments any object, in an “open-vocabulary” setting.

The real limitation of using SAFLOW, at the current state, is the speed of execution. DPT calculates the segmentation mask in 0.225 seconds, while SAFLOW takes about 10 times as long. This is also, and more importantly, a problem on the practical side for the company, since this translates into a 10-fold slowdown compared to current performance. And this represents the major bottleneck.

The Future Work chapter, Section 6.2 presents the direction for overcoming this bottleneck.

FIGURE 5.1: General scheme of the proposed pipeline.



Blue blocks represent user input, purple blocks represent outputs produced.

5.2 Proposed pipeline

At this point, the last thing to do is to present the final pipeline. Pipeline composed of all the components previously seen, analyzed and understood in this work. Figure 5 shows the general design of the proposed pipeline to better understand, on a macroscopic level, how these components interact. From the figure, one can see how the two objectives of this paper are handled. Given the input image, provided by the user, floors and walls (and the additional class, carpets) are segmented, after which, given a set of points provided by the user it is possible through the inpainting and classification network to remove the object that the points indicate to obtain the new image or to compute the same image with the object replaced with a 3D model.

Again from the image 5 we can see how the image with the object removed can be given as input to SAFLOW to re-segment the regions, so that Reality Remod can be provided with the updated information, guaranteeing the best user experience.

A more detailed representation of how the various components interact to achieve their intended goals is shown in Figure 5.2. Before beginning, it is necessary to specify that the three components of SAM, Image Encoder, Prompt Encoder, and Mask Decoder are represented multiple times within the same schema. This does not mean that there are multiple instances, but they are repeated for clarity. At this point, it is also necessary to point out that the three components of SAM are the native ones. Consequently, there is no need to differentiate SAM components in the SAFLOW flow and the Inpainting flow.

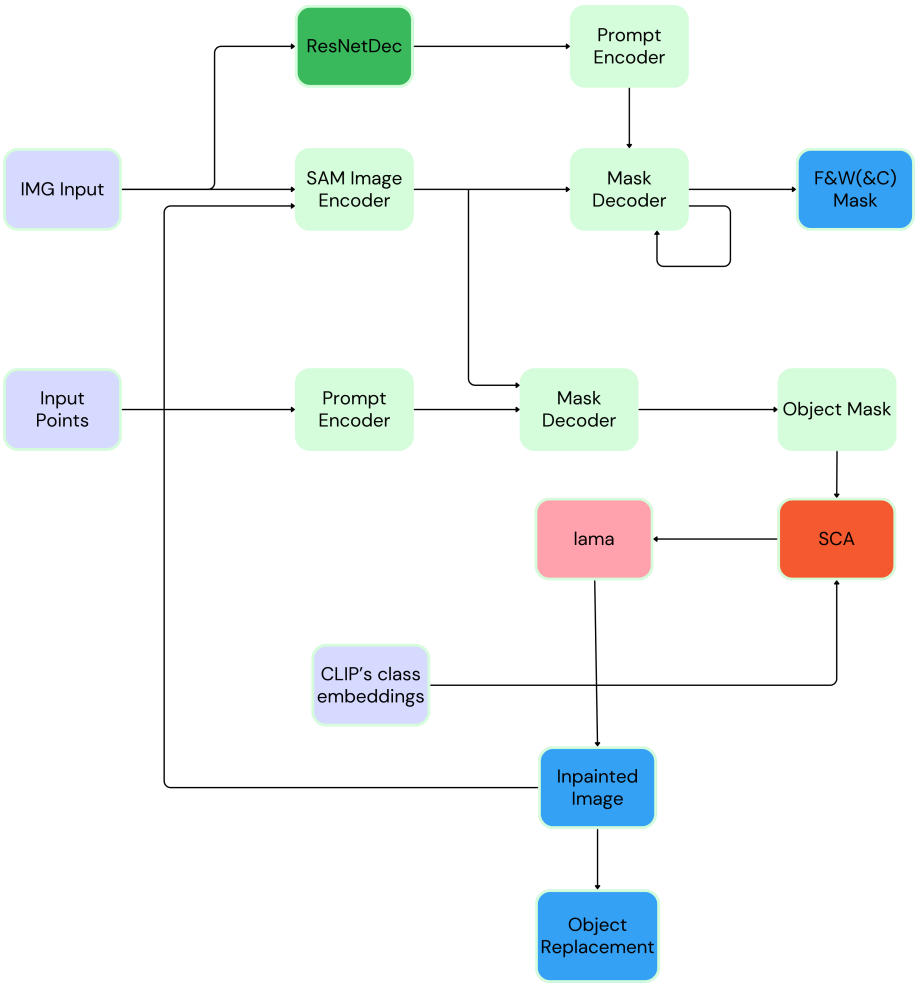
From the more detailed schema in Figure 5.2 it is possible to see how the process begins given an input image provided by the user, at which point, in parallel, both the embeddings from SAM's Image Encoder and the heatmap calculated by ResNetDec are computed, then the points from the heatmap are sampled and their embeddings computed thanks to SAM's Prompt Encoder. Once these two encodings are obtained, it is possible to compute the various region segmentation masks through the IMR approach, as seen in Section 3.3. Having computed the segmentation mask, Reality Remod, thanks also to the information obtained from the normals estimation network, can proceed with the substitution of surfaces according to the client's specifications.

This is SAFLOW, which satisfies the first requirement.

Once the image input has been provided by the user and the embeddings of SAM's Image Encoder have been computed, it can be reused to segment any other object defined by the user-supplied inputs, i.e., a set of points. Since the Image Encoder's encoding is not computed for each object and every time a segmentation is done, these interactions can be particularly fast. Once the user-supplied point encoding and the image encoding previously computed in SAFLOW are obtained, the image mask to be removed can be computed.

Before the actual removal of the object, the classification stage is carried out, for the reasons previously explained, i.e., understanding the family of the removed object to replace it with a 3D model and limiting the use of this pipeline branch to only previously selected objects. Classification will be performed following the method presented in Section 3.4, i.e. through leveraging the alignment between the SAM and CLIP space.

FIGURE 5.2: Final Pipeline proposed



Blue blocks represent user input, purple blocks represent outputs produced.

Once the class of the object to be removed is determined, if eligible, through the inpainting network chosen after the analysis conducted in Section 4.5, i.e., lama, the object will actually be removed.

At the end of this branch, to get the new segmentation mask from SAFLOW, it is necessary to start from the beginning again, then recalculate the encoding of the image - since it is changed - is necessary, but if you want to remove another object instead, it is not required to recalculate the encoding, because the mask of the new object you want to remove can be computed on the original, while you give as input to the inpainting network the image that has already been processed once.

5.2.1 Visual results

In Table 5.3, a comprehensive visual depiction of the evolutionary stages within the proposed pipeline is presented. It is imperative to underscore that the showcased results are currently in the prototypical phase of development. As such, instead of showing entirely novel coverings, the table exhibits a grid of substitutions, providing a glimpse into the transformative process underway.

The significance of Table 5.3 lies in its ability to enlighten the seamless integration of inpainting techniques within the pipeline. Despite being in the nascent stages, these visual representations serve as compelling evidence of the robustness and versatility of the proposed framework. Moreover, they highlight the efficacy of inpainting in preserving the integrity and coherence of the processed images, thereby enhancing the overall quality of the output.

TABLE 5.3: Visual examples of proposed pipeline.

Input Img	Grid	Inpainted Img	New Grid

Table 5.3 represents an input image, the grid computed to replace the covering, computed from the information supplied by SAFLOW, then the image generated by lama after the removal of an object and lastly the grid, computed again with SAFLOW but over the inpainted image.

The insights gleaned from Table 5.3 affirm the capacity of the proposed pipeline to fulfill its original objectives. However, it is crucial to recognize that this milestone is merely a stepping stone in the journey towards refinement and optimization. As emphasized earlier,

the pipeline remains in its prototype phase, leaving ample room for further enhancement and advancement.

While the results showcased in Table 5.3 are indeed promising, it is imperative to acknowledge that there are still avenues to explore and challenges to overcome. In Section 6.2, titled "Future Work", a roadmap is delineated, outlining the strategic next steps essential for transitioning the pipeline from its current prototype state to a fully realized, production-ready solution.

In essence, while Table 5.3 provides a glimpse into the capabilities of the pipeline, it is the directives outlined in Section 6.2 that pave the way for its continued evolution and eventual deployment in real-world scenarios.

Chapter 6

Conclusion and future work

6.1 Conclusion

Throughout this thesis, a comprehensive exploration of segmentation, inpainting, and classification within the realm of computer vision has been undertaken with the aim of proposing a practical framework applicable to industrial scenarios. The journey taken has explored the state of the art in these areas, providing a deep understanding necessary to develop an effective computer vision pipeline.

At the core of this proposed pipeline lie two pivotal components: ResNetDec and SAFLOW. These innovations are fundamental in releasing the full potential of segment-anything capabilities. The active development and refinement of these components have been the primary focus of this thesis, serving as the cornerstone for the envisioned framework.

Concurrently, the exploration extended to the realm of inpainting, where a thorough comparative analysis of two inpainting networks was conducted. The objective was to identify a network that not only met the stringent requirements but also seamlessly integrated within the proposed pipeline. This meticulous selection process underscores the commitment to ensuring the efficacy and robustness of the framework.

Moreover, a novel approach to classification was introduced, aiming to leverage the alignment between SAM and CLIP. While this integration holds significant promise, it is acknowledged that its realization demands a significant quantity of time and resources. Despite being in the nascent stages, preliminary experiments indicate promising prospects, signaling a potential avenue for future exploration and refinement.

In essence, this thesis culminates in the presentation of a cohesive pipeline balanced to harmoniously unite fields of computer vision. The objective has not only been to engage in an academic exercise but to develop a pragmatic solution tailored for real-world implementation within business contexts. The envisioned framework embodies the convergence of cutting-edge research and practical utility, positioning it as a valuable asset for industrial applications.

In conclusion, this thesis serves as a testament to the convergence of theoretical insights and practical ingenuity, culminating in the delineation of a versatile computer vision pipeline.

6.2 Recommendation for Future Work

Since this work is intended to be put into production by the company, several aspects need to be refined before that goal can be achieved. The work presented in this thesis represents the foundation of what is to be.

To achieve levels of both quality and worthy computational performance, that is, to be put into production, several aspects must be improved.

Firstly, the segment-anything's image encoder needs to be sped up and lightened. Following the work presented in [32], is possible to make SAM eight times faster without losing accuracy, and this means nearly reaching the performance of DPT, but maintaining all the advantages of this approach. The overlap problem, presented in 4.19, must be solved by taking into account also other factors representing, in some manner, the scene, like the estimation of the surfaces' normal. Having the direction of the surface makes it easier to classify a pixel that is associated with two different regions at the same time.

Dealing with the inpainting shadow problem is non-trivial. Leveraging morphological operation over the mask can not be a solution, especially in cluttered scenes. Another approach needs to be employed to remove also the shadow of an object.

The SAM-CLIP Alignment must be further investigated by exploiting distributed computation and better GPUs, it seems like a catchphrase these days but it is necessary to have a particular lightweight aligner, to both classify and segment in an "open-vocabulary" setting. This approach could unify two foundational models at almost no additional cost, becoming extremely useful not only in this case study but in the "open-vocabulary" segmentation scenario in general.

This topic will persist in its exploration beyond the completion of this thesis, as there remains ample scope for further investigation and analysis.

Bibliography

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Farheen Ramzan, Muhammad Usman Khan, Asim Rehmat, Sajid Iqbal, Tanzila Saba, Amjad Rehman, and Zahid Mehmood. A deep learning approach for automated diagnosis and multi-class classification of alzheimer's disease stages using resting-state fmri and residual neural networks. *Journal of Medical Systems*, 44, 12 2019. doi: 10.1007/s10916-019-1475-2.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning

transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[6] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

[7] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, October 2021.

[8] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[9] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. URL https://arxiv.org/pdf/2011.02523.pdf.

[10] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[13] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.

[14] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022.

[15] Andranik Sargsyan, Shant Navasardyan, Xingqian Xu, and Humphrey Shi. Mi-gan: A simple baseline for image inpainting on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7335–7345, October 2023.

[16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[17] 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.

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[20] Angeline Aguinaldo, Ping-Yeh Chiang, Alex Gain, Ameya Patil, Kolten Pearson, and Soheil Feizi. Compressing gans using knowledge distillation, 2019.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks, 2021.

[23] Haobo Yuan, Xiangtai Li, Chong Zhou, Yining Li, Kai Chen, and Chen Change Loy. Open-vocabulary sam: Segment and recognize twenty-thousand classes interactively, 2024.

[24] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2019.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[27] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[28] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M.

Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[29] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, dec 1996. ISSN 0098-3500. doi: 10.1145/235815.235821. URL https://doi.org/10.1145/235815.235821.

[30] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In M. Jorge Cardoso, Aasa Feragen, Ben Glocker, Ender Konukoglu, Ipek Oguz, Gozde Unal, and Tom Vercauteren, editors, *Proceedings of The 2nd International Conference on Medical Imaging with Deep Learning*, volume 102 of *Proceedings of Machine Learning Research*, pages 285–296. PMLR, 08–10 Jul 2019. URL https://proceedings.mlr.press/v102/kervadec19a.html.

[31] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

[32] Team PyTorch. Accelerating generative ai with pytorch: Segment anything, fast. https://pytorch.org/blog/accelerating-generative-ai/, 2023. Accessed: 2023-11-18.