POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

A Modular Approach for Object Segmentation and Image Inpainting

Advanced Techniques of Artificial Intelligence and Computer Vision in Industrial Environments

Alexandre Santos Silva

School of Management and Technology
Department of Informatics Engineering
Master in Computer Engineering - Mobile Computing

Leiria, October 2025

This page intentionally left blank.

POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

A Modular Approach for Object Segmentation and Image Inpainting

Advanced Techniques of Artificial Intelligence and Computer Vision in Industrial Environments

Alexandre Santos Silva

Student No. 2230329

Supervisors: José Carlos Bregieiro Ribeiro

Coordinator Professor, Department of Informatics Engineering, School of Management and Technology, Polytechnic of Leiria

Rolando Germano Lúcio Miragaia

Assistant Professor, Department of Informatics Engineering, School of Management and Technology, Polytechnic of Leiria

Rogério Luís de Carvalho Costa

Assistant Professor, Department of Informatics Engineering, Faculty of Sciences and Technology, University of Coimbra

Fernando José Mateus da Silva

Coordinator Professor, Department of Informatics Engineering, School of Management and Technology, Polytechnic of Leiria

School of Management and Technology

Department of Informatics Engineering

Master in Computer Engineering - Mobile Computing

Project Report

Leiria, October 2025

This page intentionally left blank.

A Modular Approach for Object Segmentation and Image Inpainting

Preparation of this work was facilitated by the use of the *IPLeiria-Thesis* template.

This page intentionally left blank.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors: Professors José Ribeiro, Rolando Miragaia, Rogério Costa and Fernando Silva. Their support, knowledge, guidance and mentorship were crucial for the success of the project I present in this report. I extend my appreciation to Professors Paulo Loureiro and Sílvio Mendes for providing access to their server, which was instrumental in the development of this work. Furthermore, I thank Professor Patrício Domingues for providing access to his workstation, and Professor Alexandrino Gonçalves for kindly lending a computer for model training. On a personal note, I would like to acknowledge my family: my father, Mário, my mother, Isabel, and my brother, Daniel; for their unwavering support and encouragement throughout this journey. Lastly, I would like to thank my friends for their companionship and motivation during this challenging yet rewarding experience.

I also acknowledge the *PRR - Plano para a Recuperação e Resiliência* for their financial support within the Sustainable Stone by Portugal project, Call: 2021-C05i0101-02—*agendas/alianças mobilizadoras para a reindustrialização - PRR*, Proposal: C632482988-00467016.

This page intentionally left blank.

Publications

During the development of this project, we published an article on the use of artificial intelligence in the stone manufacturing industry (A. Silva et al., 2025). The article is listed below:

- **Authors:** Alexandre Silva, Carolina Antunes, Rolando Miragaia, Rogério Costa, Fernando Silva, and José Ribeiro
- **Title:** Artificial intelligence applied to the stone manufacturing industry: A systematic literature review
- **Journal:** Computers and Electrical Engineering
- **Volume:** 128, Part A
- **Year:** 2025
- **Publisher:** Elsevier
- **ISSN:** 0045-7906
- **EISSN:** 1879-0755
- **DOI:** https://doi.org/10.1016/j.compeleceng.2025.110702
- **URL:** https://www.sciencedirect.com/science/article/pii/S0045790625006457
- **JCR Impact Factor:** 4.9 (2024)
- **CiteScore:** 10.7 (2024)
- **SJR Impact Factor:** 1.053 (2024)
- **SJR Category Rank (2024):**
  - Computer Science (miscellaneous): Q1
  - Control and Systems Engineering: Q1
  - Electrical and Electronic Engineering: Q1

This page intentionally left blank.

Abstract

In this project, we propose ModInPainTor, a modular solution for object segmentation and removal in images. ModInPainTor takes advantage of advanced models from literature to deliver high-quality image inpainting results. This solution is inspired by InPainTor, a previous project that utilized a highly-integrated Convolutional Neural Network-based model for object detection and removal but lacked a comprehensive evaluation of its performance. We conduct a thorough evaluation of the InPainTor and ModInPainTor solutions, assessing their effectiveness in anonymizing visual data while maintaining data quality. The evaluation involves quantitative metrics to measure the quality of the results and a qualitative analysis to evaluate the visual quality of the anonymized images. Furthermore, we investigate the impact of object segmentation accuracy on the quality of the anonymized images. Our findings indicate that ModInPainTor significantly improves the visual quality of anonymized images compared to InPainTor, albeit with increased computational requirements. We find that the proposed solution could be effectively employed in industrial settings where anonymization quality is prioritized over computational efficiency.

Keywords: Anonymization, Image Inpainting, Object Segmentation, Object Removal.

This page intentionally left blank.

AI Acknowledgement

I acknowledge the use of Gemini to assist in writing, formatting and refining this project report, including aspects of grammar, punctuation, and vocabulary. In some cases, Gemini was also used to generate code snippets or aid in debugging issues in Python and LaTeX. I also acknowledge the use of GitHub Copilot, with GPT-4o and GPT-4.1 models, to assist in writing code snippets in Python and LaTeX.

This page intentionally left blank.

Contents

This page intentionally left blank.

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

Acronyms

Adam Adaptive Moment Estimation. (*p. 26*)
AdamW Adam with Weight Decay. (*p. 26, 42*)
AP Average Precision. (*p. 27*)
ASPP Atrous Spatial Pyramid Pooling. (*p. 9*)

BCE Binary Cross Entropy. (*p. 24, 26, 42*)
BiSeNet Bilateral Segmentation Network. (*p. 37, 38*)

CAM Contextual Attention Module. (*p. 29, 35*)
CGAN Conditional Generative Adversarial Network. (*p. 16*)
CNN Convolutional Neural Network. (*p. xii, 1, 3–8, 10, 12, 13, 15, 20, 21, 29, 34*)
COCO Common Objects in Context. (*p. 22, 23, 32, 37–42, 44, 47, 48, 56, 67, 85, 86*)
CoModGAN Conditional Modulation Generative Adversarial Network. (*p. 18, 19*)
CRF Conditional Random Fields. (*p. 9*)
CV Computer Vision. (*p. 3, 7, 21*)

DCGAN Deep Convolutional Generative Adversarial Network. (*p. 16*)
DETR DEtection TRansformer. (*p. 13*)
DL Deep Learning. (*p. 3, 7, 14, 15, 26, 34, 40, 85*)
DNNAM Deep Neural Networks and Attention Mechanism. (*p. 19*)
DSR Design Science Research. (*p. 31, 85*)

ENet Efficient Neural Network. (*p. xiii–xxi, 33, 34, 37, 38, 40, 42–44, 46–64, 66–82, 85, 86*)

FCL Fully Connected Layer. (*p. xii, 5–7, 10, 11, 16, 35*)
FCN Fully Convolutional Networks. (*p. 7, 11*)
FFC Fast Fourier Convolution. (*p. 20*)
FFN Feed-Forward Network. (*p. 21*)
FID Fréchet Inception Distance. (*p. 28, 29*)
FPN Feature Pyramid Network. (*p. 12, 13*)

GAN Generative Adversarial Networks. (*p. 15–20, 30, 34, 35, 83, 85*)

GDPR General Data Protection Regulation. (*p. 1*)
GFLOPS Giga Floating Point Operations Per Second. (*p. 46, 48, 82, 83*)
GLCIC Globally and Locally Consistent Image Completion. (*p. 17, 29*)
GPU Graphics Processing Unit. (*p. 6, 40*)

ILSVRC ImageNet Large Scale Visual Recognition Challenge. (*p. 6*)
IoU Intersection over Union. (*p. 25–27, 38*)
IPS Images Per Second. (*p. 47–49, 65–67, 82, 83, 86*)

K Key. (*p. 21*)

LPIPS Learned Perceptual Image Patch Similarity. (*p. 28, 29, 83*)

Mask R-CNN Mask Region-based Convolutional Neural Network. (*p. 11–13*)
MFMAM Multi-scale Feature Module with Attention Module. (*p. 19*)
mIoU mean Intersection over Union. (*p. 27, 38, 42–44, 47–49, 81, 85*)
MSE Mean Squared Error. (*p. 28, 44, 45*)

NLP Natural Language Processing. (*p. 21*)

PConv Partial Convolution. (*p. 29*)
PDE Partial Differential Equations. (*p. 14*)
PEPSI Parallel Extended-decoder Path for Semantic Inpainting. (*p. xiii, xvii–xxi, xxiii, 18, 34–36, 39–41, 45, 46, 66, 67, 69–83, 85*)
PQ Panoptic Quality. (*p. 27, 28*)
PSNR Peak Signal-to-Noise Ratio. (*p. xvii, 28, 32, 65–68, 75, 76, 82, 83, 85, 86*)

Q Query. (*p. 21*)

R-CNN Region-based Convolutional Neural Network. (*p. 10–12*)
RED Region Ensemble Discriminator. (*p. 35*)
ReLU Rectified Linear Unit. (*p. 4, 6*)
ResNet Residual Networks. (*p. 6, 10, 13*)
RoI Region of Interest. (*p. 10–12*)
RORD Real-world Object Removal Dataset. (*p. xvii–xx, xxiii, 2, 23, 34, 35, 41, 44, 45, 65–77, 82, 85, 86*)
RPN Region Proposal Network. (*p. 11*)
RPVD RORD-Person Validation Dataset. (*p. xvii, xxiii, 2, 41, 65, 67, 68, 77, 85, 86*)

SGD Stochastic Gradient Descent. (*p. 26*)
SSIM Structural Similarity Index Measure. (*p. xvii, 28, 29, 32, 65–68, 70, 75, 76, 81–83, 85, 86*)
SUN Scene UNderstanding. (*p. 22*)

V Value. (*p. 21*)

VGGNet Visual Geometry Group Network. (*p. 6, 10*)

VOC Visual Object Classes. (*p. 22*)

WGAN Wasserstein Generative Adversarial Network. (*p. 16*)

WGAN-GP Wasserstein Generative Adversarial Network with Gradient Penalty. (*p. 16*)

This page intentionally left blank.

1 Introduction

Thanks to today's quality control and automation systems, industries today heavily rely on visual data. However, this data can often contain sensitive and personally identifiable information that needs to be protected. This has made it a legal and ethical necessity for industries to implement robust anonymization protocols, especially with the rise of regulations such as the General Data Protection Regulation (GDPR).

As such, there is a pressing need for effective and efficient anonymization techniques that can handle the vast amounts of visual data generated daily. Traditional methods of anonymization, such as blurring or pixelation, often compromise the quality and utility of the data, making it less useful for analysis and decision-making. Furthermore, manual anonymization processes are time-consuming and prone to human error, making them impractical for large-scale applications. The central problem addressed in this project is the development of automated anonymization techniques that can effectively obscure sensitive information while preserving the overall integrity and usability of the visual data.

This work builds upon a prior project, InPainTor (Ribeiro, 2024), which used a highly-integrated CNN-based model for object detection and removal. Its primary goal was to be computationally lightweight. However, this project lacked a comprehensive evaluation of the model's performance. This project aims to fill that gap by conducting a thorough evaluation of the InPainTor model, assessing its effectiveness in removing unwanted objects from images. The evaluation will involve quantitative metrics to measure the degree of anonymization and qualitative assessments to evaluate the visual quality of the anonymized images.

Inspired by InPainTor, we propose a new solution for object removal in images, taking advantage of more robust and recent models. We name this solution ModInPainTor. The primary goal is to improve the visual quality of the anonymized images, even if at the expense of computational efficiency. To achieve this, we develop a modular architecture that allows for easy integration of different image segmentation and inpainting models. This modularity allows for future improvements to be easily incorporated, as long as they follow the same format. The proposed solution will be evaluated against

the InPainTor model to determine its effectiveness and efficiency in anonymizing visual data. The evaluation will consider various scenarios and types of visual data to ensure the robustness of the proposed solution.

For this project, we aim to answer the following research questions:

- How effective is the InPainTor model in anonymizing visual data?
- What are the strengths and limitations of the InPainTor model in various anonymization scenarios?
- How does the proposed solution, ModInPainTor, compare to the InPainTor model in terms of visual quality and computational efficiency?
- What is the impact of the accuracy of object segmentation on the quality of the anonymized images?
- How much can the performance of a segmentation model from literature be optimized to improve its accuracy?
- How can the proposed solution be optimized for real-time applications in industrial settings?

Our main contributions include a comprehensive evaluation of the InPainTor model, with the identification of its strengths and limitations. We propose ModInPainTor, a modular and flexible solution for object segmentation and removal, using more advanced models from literature. We evaluate ModInPainTor against InPainTor, both quantitatively and qualitatively, and assess its performance in object removal tasks. To analyze its performance on anonymizing people, we create a new dataset based on RORD, which we name "RPVD". Additionally, we research the impact of object segmentation accuracy on anonymization quality and work on optimizing a segmentation model from literature to enhance its performance.

The structure of this project report is as follows: In Chapter 2, we provide a review of the relevant literature and background information on object detection, segmentation, and removal techniques. Chapter 3 dives into the research methodology we adopted, and details the architecture of the proposed solution, ModInPainTor, including the selection and integration of models for segmentation and inpainting. In Chapter 4, we outline the experimental setup, including the preparation of the datasets, model training and optimization, and present the results of our experiments, comparing ModInPainTor with InPainTor and analyzing the impact of segmentation accuracy on anonymization quality. Finally, in Chapter 5, we summarize our findings, discuss the implications of our work, and suggest directions for future research.

2 Background

In this Chapter, we will introduce core concepts, key architectures, training approaches, datasets and metrics on image segmentation and inpainting solutions.

2.1 Deep Learning

Deep Learning (DL) is a subset of machine learning that uses neural networks with multiple layers to learn and represent data. It has been successfully applied to various fields, including Computer Vision (CV) (Chai et al., 2021), natural language processing (Lauriola et al., 2022) and speech recognition (Ahlawat et al., 2025).

2.1.1 Core Concepts

DL-based image segmentation typically takes advantage of some core concepts, such as CNNs, encoder-decoder architectures and skip connections (Minaee et al., 2021).

CNNs are some of the most widely used techniques for image segmentation. They typically take advantage of three types of layers: convolutional, nonlinear activation and pooling layers (Minaee et al., 2021).

Convolutional layers apply ”filters” (or ”kernels”), which slide across the input image. In the convolution process, element-wise multiplication is performed between the filter and the portion of the image it covers. The multiplication results are summed into one value, which represents how strongly a certain feature is represented in the area covered by the filter. These filters contain weights, which are learned in the training process. Each filter is used to extract different features, such as edges, textures or shapes, and a convolutional layer typically has multiple filters. Figure 2.1 illustrates the convolution operation.

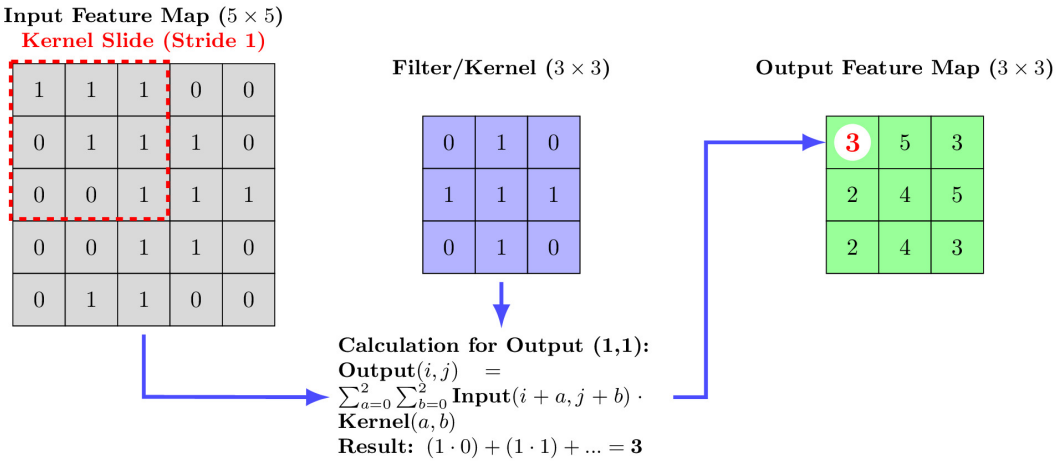

Figure 2.1: *Illustration of the convolution operation, where a 3x3 filter is applied to a 5x5 input image, resulting in a 3x3 feature map.*

Nonlinear activation functions, such as Rectified Linear Unit (ReLU), are applied after each convolutional layer. By introducing non-linearity to the model, it is able to learn more complex patterns.

Pooling layers reduce the resolution of the feature maps by sliding a window across each feature map, similarly to how convolutional layers operate. However, the windows differ from filters by not containing learnable weights. Instead, a specific aggregation function is applied to each pooling window, and a single value is taken from that region. The most common pooling operations are max pooling and average pooling. In max pooling, the maximum value within each window is selected, whereas in average pooling, the mean of all values within the window is calculated. Using pooling layers comes with several advantages:

- Downsample feature maps – decreases the number of parameters and computations in subsequent layers of the networks;
- Translation invariance – small shifts or distortions in the input image do not drastically change the output of the pooling layer; increases robustness and allows the model to better generalize to new data;
- Feature abstraction – summarises the information in a region, extracting the most important features and creating a more abstract representation of the input;
- Prevent Overfitting – as it reduces the number of parameters, overfitting the model to training data becomes more difficult.

CNNs are particularly efficient as they use parameter sharing. The same filter is applied across different parts of the input, which reduces the number of parameters the network needs to learn.

Encoder-decoder architectures are used on most image segmentation solutions. As the name implies, they are divided into two parts: an encoder – typically a CNN, used for extracting features from an input image – and a decoder – which upsamples the

feature maps to create a segmentation mask at a higher resolution, normally equal to the input image's size.

Skip connections, often known as residual connections, are used in many encoder-decoder networks. They connect feature maps from the encoder directly to the decoder, allowing information and gradients to be used in non-consecutive layers. As such, the output of a layer is not only dependent on its own input, but it can also incorporate the input from an earlier layer. The use of skip connections in these networks aids in preventing vanishing gradients – a common problem where the gradients used to update model weights during the training process become very small as they are backpropagated through many layers. They are also particularly useful for preserving details, as information is lost through downsampling operations such as convolution and pooling. Skip connections allow high-resolution features from early downsampling stages to be carried over to later layers, such as in the decoder, so that image reconstruction can occur with minimal information loss. By minimising these problems, the models can be more effectively trained, leading to better performance, stability, faster convergence and more robustness.

FCLs typically come at the end of the CNN architecture. They transform the features extracted by earlier layers into a final output. They are mostly used in image classification or regression tasks, where they output class probabilities or numerical values, respectively. They are not commonly used for image segmentation, as they discard spatial information. These layers function as traditional neural networks, where each neuron is connected to every single neuron from the previous layer. As such, the previous layer must be flattened into one single vector. When dealing with image segmentation tasks, and more specifically in semantic image segmentation, the output of a network should be a mask where each pixel represents a probability of it belonging to a class. Some early segmentation architectures experimented with FC layers, but they required specific processing techniques such as patch-based classification or sliding windows. These techniques were computationally heavy and resulted in inconsistent segmentation boundaries. An example of a CNN with an FCL can be seen in Figure 2.2.

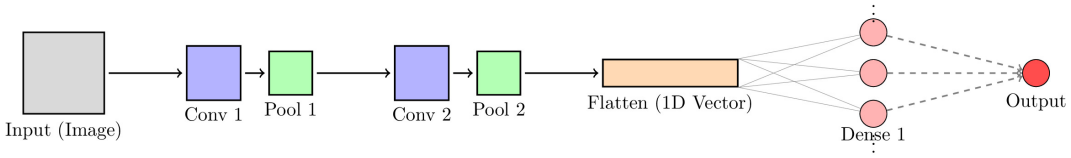

Figure 2.2: *Example of a CNN architecture with a FCL at the end, used for image classification.*

2.1.2 Encoders/Backbones

In encoder-decoder architectures, it is common to use other CNN architectures as encoders (or backbones). These CNNs were not specifically designed for image segmen-

tation, but they show good performance at feature extraction. Some of them include AlexNet (Krizhevsky et al., 2017), GoogLeNet/Inception (Szegedy et al., 2014), Visual Geometry Group Network (VGGNet) (Simonyan et al., 2015) and Residual Networks (ResNet) (He et al., 2015).

AlexNet (Krizhevsky et al., 2017) was developed in 2011 and showed groundbreaking performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 (Russakovsky et al., 2015). Its performance is attributed to the use of deep CNNs, ReLU activation functions for non-linearity and Graphics Processing Unit (GPU) acceleration. The model was trained using two Nvidia GTX580 GPUs, each with 3GB of video memory. This was a pivotal architecture, which laid the foundation for later segmentation encoders.

In 2014, researchers at Google developed the GoogLeNet architecture, later renamed to Inception (Szegedy et al., 2014). The first version won the ILSVRC 2014 (Russakovsky et al., 2015). It introduced inception modules, which performed multiple convolutions with different kernel sizes (1x1, 3x3 and 5x5) and max pooling in parallel which resulted in an output after concatenation. While it is a deep network, it was able to be more computationally efficient than AlexNet while performing better. Another modification when compared to AlexNet is the use of global average pooling for the final layers instead of FCLs. To combat the vanishing gradient problem, GoogLeNet takes advantage of auxiliary classifiers at 1/3 and 2/3 of the network, which provide intermediate loss values which are then used to propagate gradients back through the network more effectively. These classifiers are only used during the training process, being discarded for inference.

VGGNet was released in that same year (Simonyan et al., 2015). It took inspiration from AlexNet (Krizhevsky et al., 2017), but explored the use of small 3x3 convolutional filters throughout the network, whereas AlexNet and other earlier CNNs used larger filters. VGGNet used several generic modules in series, comprising convolutional modules with 3x3 kernels, stride of 1 and ReLU activation, followed by max pooling layers with 2x2 kernels and stride of 2 to downsample feature maps, and three FCLs, the last one with 1000 channels corresponding to the 1000 classes in ImageNet. Finally, a SoftMax activation layer provides class probability. Several configurations of the VGGNet were released, the most commmonly used being VGG-16 and VGG-19, with 13 and 16 convolutional layers, respectively, and 3 FCLs. While its performance was slightly lower than Inception's performance, it still reached state-of-the-art results, while employing a significantly simpler architecture. It was widely used as an encoder backbone in early segmentation models.

In 2015, Microsoft researchers introduced ResNet (He et al., 2015), which won the ILSVRC (Russakovsky et al., 2015) that year. Through the use of residual blocks, this network effectively addresses the degradation problem in very deep neural networks. This problem happens when a network uses many layers, resulting in worse training accuracy. This was due to the difficulty of optimising very deep networks, in part due to the vanishing/exploding gradient problem, where as gradients backpropagate

through many layers, they either become too small (vanishing) or too large (exploding). The residual blocks incorporate skip connections, which add the input of the block to its output. As such, the residual block does not have to learn an unreferenced function H(x), which optimally results in the desired output for that layer, but a residual function F(x) = H(x) - x. This is the difference needed to go from x to H(x), and the goal of the residual block is to minimise this value as much as possible. The output of the block becomes F(x) + x, which is equal to H(x). The "+ x" is the skip connection, which provides a shortcut for x, allowing gradients to flow more easily through the network during backpropagation. This results in a more stable training process and faster convergence.

2.2 Image Segmentation

Image segmentation is a fundamental task in CV which is used for identifying objects in images. The result is usually a mask where each pixel represents a class of objects. The classification of each pixel takes several characteristics into account, such as color, texture or shape of its surrounding area.

There are three types of segmentation: semantic, instance and panoptic. In semantic segmentation, each pixel is assigned a class. There is no separation between objects, which means that in an image where, for example, three instances of an object are found, there is no differentiation between them. In instance segmentation, each object is identified and assigned a segmentation mask. Panoptic segmentation is a more recent technique, introduced in 2019 (Kirillov et al., 2019b), and combines both semantic and instance approaches by assigning a class to each pixel, but also instance IDs to each object.

While today the most popular methods of image segmentation are based on DL, traditional algorithms used to be the norm. Traditional images segmentation techniques include thresholding, region-based methods (such as region growing and region splitting and merging), edge-based methods, clustering-based methods, graph-based methods, watershed transform and active contours (Y. Yu et al., 2023).

2.2.1 Semantic Segmentation Architectures

Semantic segmentation architectures build on these encoders, with the objective of performing pixel-wise classification.

One important step towards better performing architectures was the introduction of the concept of Fully Convolutional Networks (FCN)s in 2014 (Long et al., 2015). These networks allowed CNNs to perform pixel-wise classification by replacing FCLs with convolutional layers. These networks often use backbones such as VGG or AlexNet for feature extraction, but replace the FCLs with convolutional layers.

Another pivotal architecture for semantic image segmentation tasks is U-Net (Ronneberger et al., 2015). It was developed in 2015 after the introduction of FCNs, on

which it is based, and uses an encoder-decoder architecture. The encoder, also known as the contracting path, progressively reduces spatial dimensions, while increasing the number of feature channels and, thus, feature information. The decoder, also known as the expanding path, reconstructs the spatial resolution of the image while localising features learned by the encoder. At each step of the decoder, the upsampled feature map is concatenated with the corresponding feature map from the encoder at the same resolution. By using these skip connections, the decoder is able to connect the contextual information with precise spatial details, which results in accurate pixel-wise segmentation. Figure 2.3 illustrates the U-Net architecture.

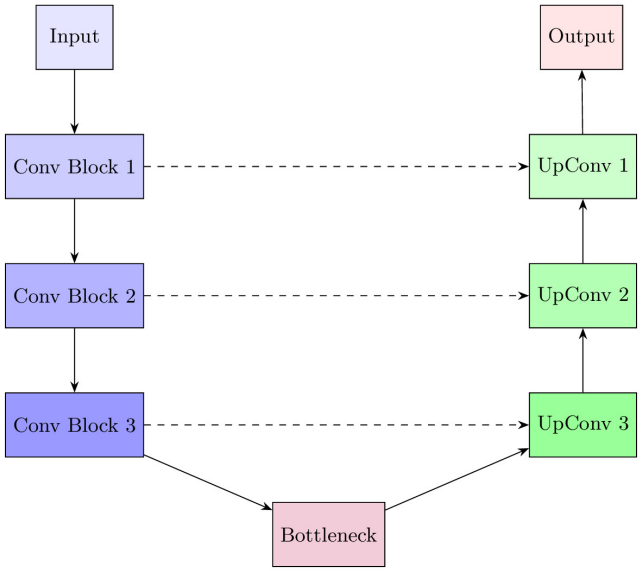

Figure 2.3: *U-Net architecture, featuring an encoder-decoder structure with skip connections between corresponding layers in the encoder and decoder paths. Adapted from* (*Ronneberger et al., 2015*).

DeepLab models introduced several pivotal changes. One core concept used by these models is Atrous Convolution, also known as Dilated Convolution, which was introduced in DeepLabV1 (L.-C. Chen et al., 2016). In traditional CNNs, convolutions reduce the spatial resolution of the feature maps. With Atrous convolution, filters have empty spaces between receptive fields. As such, the network can capture wider contexts without losing spatial details. Figure 2.4 illustrates the difference between standard and atrous convolutions.

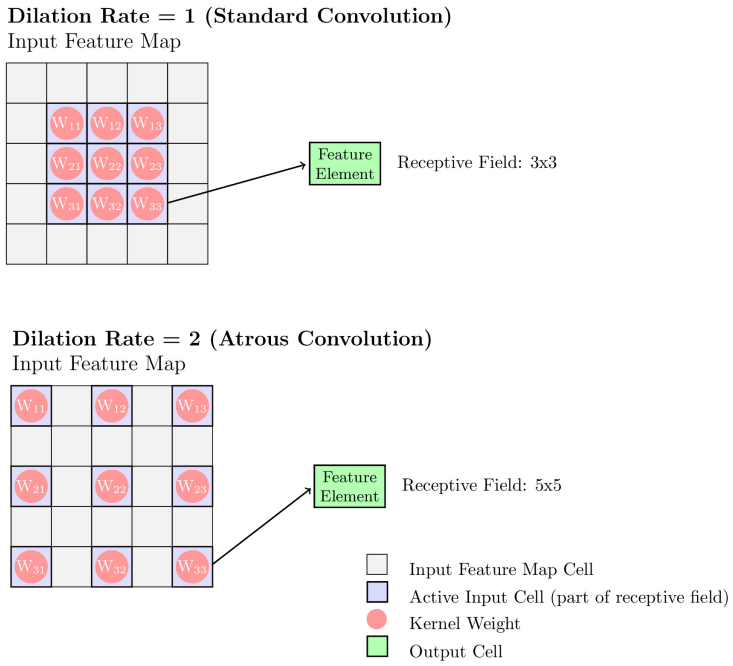

Figure 2.4: *Comparison between standard convolution (left) and atrous convolution (right) with a dilation rate of 2. In atrous convolution, the filter has spaces between its receptive fields, allowing it to capture a larger context without reducing spatial resolution.*

DeepLabV2 (L.-C. Chen et al., 2017a) introduced Atrous Spatial Pyramid Pooling (ASPP), which applies atrous convolutions at various dilation rates (the space between receptive fields) in parallel, allowing the network to capture context at various scales. These convolutions are then fused together. Figure 2.5 illustrates the ASPP module.

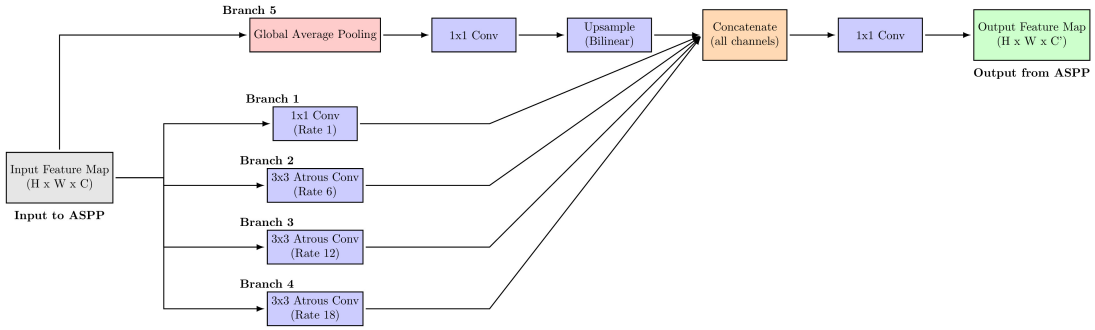

Figure 2.5: *Atrous Spatial Pyramid Pooling (ASPP) module, which applies atrous convolutions at multiple dilation rates in parallel to capture multi-scale context. The outputs are then fused together.*

The first two versions of DeepLab use Fully Connected Conditional Random Fields (CRF)s as a post-processing step. CRFs refine segmented object's boundaries by identifying neighboring pixels with similar characteristics as being likely to belong to the same label. As the newer versions of DeepLab introduced changes to its architecture, the use of CRFs was no longer required. These versions, DeepLabV3 (L.-C. Chen et al., 2017b) and DeepLab V3+ (L.-C. Chen et al., 2018), moved to an encoder-decoder

architecture, similar to U-Net (Ronneberger et al., 2015), while taking advantage of atrous convolutions. The encoder uses a backbone such as ResNet (He et al., 2015) or MobileNet (Howard et al., 2017), along with atrous convolutional layers. The decoder recovers the spatial resolution by upsampling the feature maps back to the input image's original size, while incorporating features from the encoder, just like U-Net's skip connections.

While semantic segmentation provides a pixel-wise classification for every object in an image, some applications – such as robotic grasping, autonomous driving and medical image analysis – require the identification of individual objects belonging to the same class.

2.2.2 Instance Segmentation Architectures

Instance segmentation architectures typically build upon object detection architectures by adding a segmentation head. They often process images in two stages: first, the objects are detected; then, pixel-wise segmentation is performed for each object, resulting in masks. As such, where semantic segmentation architectures built on image classification CNNs, instance segmentation architectures take advantage of object detection backbones such as the Region-based Convolutional Neural Network (R-CNN) family of networks.

R-CNN (Girshick et al., 2014) was originally developed in 2014, using AlexNet as its backbone. Later on, R-CNN was modified to allow the use of other image classification networks, such as VGGNet (Simonyan et al., 2015), thus improving performance. The backbone network is used to extract a fixed-length feature vector for each region proposal generated by an external algorithm, such as selective search. This solution was computationally expensive.

Fast R-CNN (Girshick, 2015) was the first architecture to improve on R-CNN's structure. Instead of running each region proposal through the backbone network, the entire image is now processed through the backbone only once. Then, a Region of Interest (RoI) pooling layer projects the region proposals obtained externally onto the feature map and extracts the fixed-length feature vectors for each proposal. These are fed into FCLs for classification and bounding box regression. Figure 2.6 illustrates the Fast R-CNN architecture.

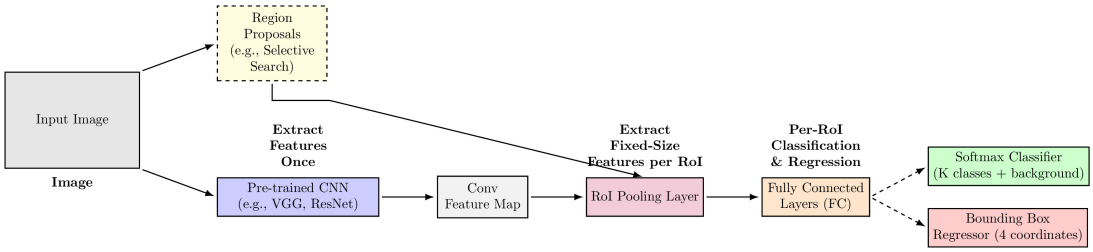

Figure 2.6: *Fast R-CNN architecture, where the entire image is processed through a backbone CNN to produce a feature map. Region proposals are then projected onto this feature map using an RoI pooling layer to extract fixed-length feature vectors for each proposal, which are subsequently fed into fully connected layers for classification and bounding box regression. Adapted from (Girshick, 2015).*

Faster R-CNN (Ren et al., 2016) further improved efficiency by generating region proposal within the network itself. The backbone processes the entire image, outputting a feature map, which is then fed into the Region Proposal Network (RPN). The proposals generated by the RPN, along with the feature map, pass through the RoI pooling layer to extract the feature vectors. As in Fast R-CNN, these vectors are fed into FCLs for classification and bounding box regression.

Mask Region-based Convolutional Neural Network (Mask R-CNN) (He et al., 2018), introduced in 2017, is the foundational instance segmentation architecture. It builds upon Faster R-CNN, an object detection network, and adds a third branch in parallel with the classification and bounding box regression branches. This new branch is an FCN, and it is used to predict a binary mask for each RoI. Figure 2.7 illustrates the Mask R-CNN architecture.

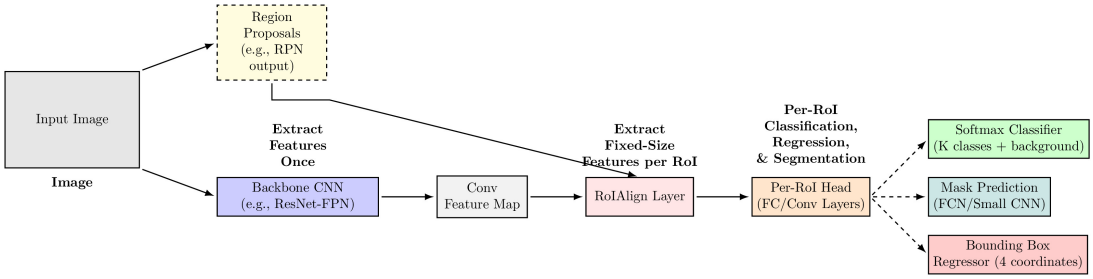

Figure 2.7: *Mask R-CNN architecture, which extends Faster R-CNN by adding a parallel branch for predicting binary masks for each Region of Interest (RoI). The network consists of a backbone CNN for feature extraction, a Region Proposal Network (RPN) for generating region proposals, and three parallel branches for classification, bounding box regression, and mask prediction. Adapted from (He et al., 2018).*

Furthermore, this architecture replaces RoI pooling with RoIAlign. These two layers serve the same purpose: to extract a fixed-size feature map from a RoI in a larger feature map. The problem is that RoI pooling rounds the RoI floating-point coordinates to integers, meaning that the RoI must align with the discrete grid of pixels on the feature map. The RoI is then divided into bins, and their dimensions are also rounded to integer values. A pooling operation, typically max pooling, is applied to each bin, where a single value is selected. The output is the fixed-size feature map. As both RoI

and bin boundary coordinates are quantized, some misalignment between the original RoI and the features extracted from the feature map is introduced. While this is acceptable for object detection tasks, it can lead to blurry or misaligned masks in instance segmentation tasks.

RoIAlign introduces some critical changes to obtain better segmentation performance. First, both RoI and bin boundary coordinates are kept as floating-point values. Then, instead of performing max pooling in each bin, it uses bilinear interpolation. In each bin, a few fixed points within its boundaries are sampled and, for each sample point, a feature value is calculated by interpolating the values of the four nearest integer pixels on the feature map. These values are aggregated, usually by average or max pooling, to get the final bin value.

Feature Pyramid Networks (FPNs) (Lin et al., 2017) are often combined with Faster R-CNN and Mask R-CNN. Traditional CNNs extract features through a bottom-up pathway. This means that at each layer, the spatial resolution of the feature maps decreases, while semantic richness increases. As such, deeper layers have a low resolution, useful for detecting large objects, but they lack precise spatial information, making it difficult to localise small objects. Conversely, shallow layers have a high resolution, which makes it easier to detect where small objects are located, but as they lack semantic information – such edges and textures – they have more difficulty in assessing what the objects really are. Figure 2.8 illustrates the concept of a feature pyramid.

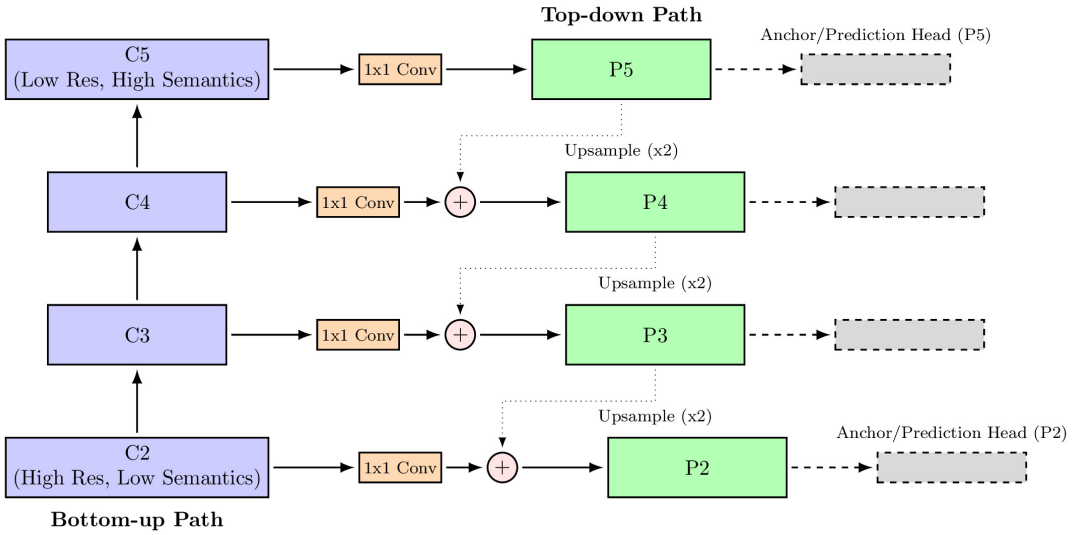

Figure 2.8: *Feature Pyramid Network (FPN) architecture, which creates a feature pyramid from a backbone CNN by adding a top-down pathway for upsampling and lateral connections for merging high-level semantic information with low-level spatial information. Adapted from (Lin et al., 2017).*

By creating a feature pyramid, it is possible to preserve high spatial resolution and rich semantics at all levels. A backbone CNN produces several feature maps as it goes into deeper layers, in the bottom-up pathway. Each feature map contains more semantic context, but loses spatial resolution. The Feature Pyramid Network (FPN) features a

top-down pathway, where it upsamples each feature map from the backbone to match the spatial resolution of the next layer, starting from the deepest layer. The upsampled feature map retains the semantic context of the original, lower resolution feature map. Each upsampled feature map is combined with the corresponding feature map from the bottom-up pathway by a lateral connection, which injects fine-grained spatial information back into the upsampled feature map. Each feature map will have stronger semantics and good spatial localisation.

In Mask R-CNN, the FPN can integrate a backbone CNN, such as ResNet (He et al., 2015), as its bottom-up pathway. The prediction heads used for classification, bounding box regression and mask prediction are then applied to each level of the pyramid, resulting in overall improved performance, as small objects are detected by using the higher-resolution feature maps and large objects by using lower-resolution maps.

Since Mask R-CNN's introduction in 2017, other instance segmentation architectures have been developed, surpassing its performance. Some of the more notable works include YOLACT (Bolya et al., 2019), which introduced a real-time instance segmentation architecture, and Mask2Former (Cheng et al., 2022), a transformer-based architecture that can perform instance, semantic and panoptic segmentation. Mask R-CNN is still widely used as a benchmark for instance segmentation tasks.

2.2.3 Panoptic Segmentation Architectures

Panoptic segmentation architectures often combine semantic and instance segmentation networks with a merging strategy. It was introduced in 2019 by Kirillov et al. (Kirillov et al., 2019b), who also proposed a baseline architecture, PanopticFPN (Kirillov et al., 2019a).

PanopticFPN pairs the Mask R-CNN with FPN architecture for instance segmentation with a custom semantic segmentation branch, in parallel with the region-based instance segmentation branch. The outputs from the instance and semantic segmentation branches are combined to create the panoptic segmentation map. Where instances overlap with the semantic predictions, the instances are prioritised.

Mask2Former (Cheng et al., 2022) adopts a different architecture, comprising a backbone, a pixel decoder and a transformer decoder with masked attention. This architecture results directly in a panoptic segmentation map. It is based on MaskFormer (Cheng et al., 2021) and DEtection TRansformer (DETR) (Carion et al., 2020), two earlier transformer-based models. The backbone is used to extract feature maps, which are then processed by the pixel decoder, which upsamples them to create high-resolution feature maps. The transformer decoder takes a set of object queries as its input, which are used to look for potential segments in the image. For each object query, the masked attention mechanism focuses on the foreground region of the predicted mask, thus extracting precise features for each segment. It also processes features from different scales of the pixel decoder's output, allowing to effectively segment objects regardless of their size. The transformer outputs, for each object query, a class label, a mask em-

bedding and a binary mask, result of the dot-product between the mask embedding and the features from the pixel decoder. The binary masks are combined and assigned instance IDs for ”things” and semantic labels for ”stuff”.

2.3 Object Removal/Inpainting

Object removal is a task which comprises the identification of unwanted objects from am image and its replacement with content that blends with its surroundings. At its core, it is a specialised application of image inpainting.

Image inpainting is the process of filling in missing or damaged parts of an image. The goal is to make the filled-in region visually indistinguishable from the surrounding content, maintaining the overall visual structure of the original image. It is highly challenging problem, as there are many possible solutions for each situation.

While image inpainting typically deals with missing or corrupted regions of an image, object removal specifically targets the elimination of unwanted objects. This process requires two stages: segmentation of the object, and its replacement with plausible content.

Unlike in image segmentation tasks, where there is a clear output that needs to be obtained, there is no specific output for a given input. This poses a highly challenging task, where lots of variables must be taken into account. The algorithm must be able to understand the scene and infer what should be in the missing area. It is crucial that the process of replicating complex textures and repetitive patterns does not introduce noticeable artifacts. Maintaining structural coherence is another issue, as lines, curves and other geometric structures must connect with the surrounding context without introducing visual discontinuities. Furthermore, the synthesised content must fit in the image’s lighting conditions. Ideally, shadows and reflections cast by the removed object should be tackled.

2.3.1 Traditional Methods

Before DL-based techniques, some traditional methods were used for image inpainting and object removal tasks. There are two main categories of traditional methods: patch-based and diffusion-based.

Patch-based methods work by searching for similar patches in the known part of the image and blending them into the missing areas. They often struggle with large missing regions, highly textured areas or when there is simply not enough context in the image. They can lead to repetitive textures, visual discontinuities and fail to reconstruct semantic content.

Diffusion-based methods propagate information from the boundary between known and missing areas into the unknown region of the image, using Partial Differential Equations (PDE)s. While they are effective for small and smooth missing regions, they tend to blur finer details and cannot create new structures or textures. The results are

often blurred, and complex missing areas tend to be unconvincing.

While these traditional methods were functional, they often required significant manual intervention and were very computationally expensive for high-resolution images. They also lacked the ability to understand the image context and generate new, plausible content. It was clear that further research was required to find better-suited approaches for image inpainting and object removal.

2.3.2 Deep Learning Methods

The modern approach to image inpainting is to use DL networks. Generative models are able to understand surrounding context and infer missing information, such as colors, textures, patterns, lighting, shadows and perspective. As such, these models can produce better results, often indistinguishable from unedited images.

In the early days of DL image inpainting, research was primarily focused on using CNNs. Their ability for feature extraction and pattern recognition, essential for image inpainting, made them a suitable approach.

Autoencoders were another common approach. These comprise an encoder, which compresses input data into a lower-resolution latent representation of itself, and a decoder, which reconstructs this representation while filling in the missing regions of the image. Basic autoencoders still struggled with generating realistic textures and varied content, often outputting blurry results.

Generative Adversarial Networks

A significant leap was made with the introduction of Generative Adversarial Networks (GAN)s (I. J. Goodfellow et al., 2014). These networks feature two mechanisms, the generator and the discriminator, which work together to push the model toward more convincing results. The generator is a neural network, often an autoencoder or a variant of U-Net, which takes an image with one or more missing regions and a mask as input and tries to generate a complete, plausible image. On the other hand, the discriminator, another neural network, tries to assess if the image is real or fake. The two networks are trained simultaneously: the generator tries to "trick" the discriminator into identifying its images as real, and the discriminator tries to accurately identify the generator's images as being fake. Figure 2.9 illustrates the architecture of a GAN.

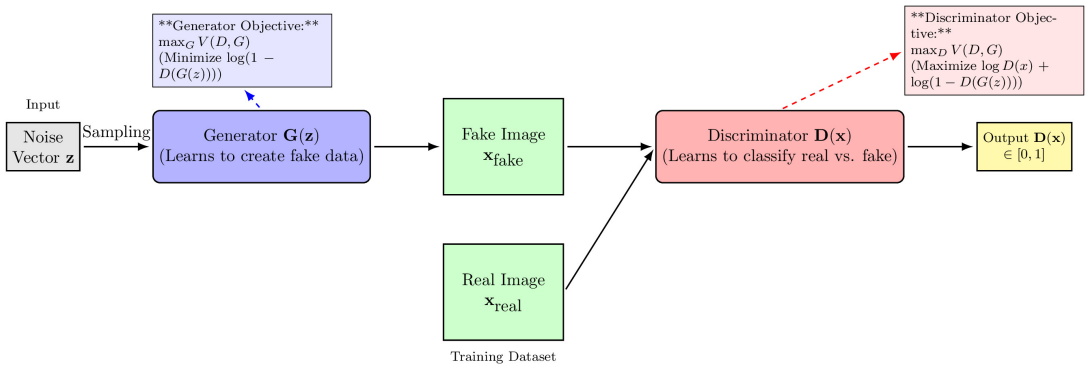

Figure 2.9: *Generative Adversarial Network (GAN) architecture, featuring a generator that creates fake images from noise and a discriminator that classifies images as real or fake. The two networks are trained in opposition to each other, with the generator trying to fool the discriminator and the discriminator trying to correctly identify real and fake images. Adapted from (I. J. Goodfellow et al., 2014).*

Some early works on GANs focused on adapting the original architecture or high-resolution image tasks. Two pivotal works in this regard were Deep Convolutional Generative Adversarial Network (DCGAN) (Radford et al., 2016) and Conditional Generative Adversarial Network (CGAN) (Mirza et al., 2014). DCGAN replaced the FCLs in the original architecture with a deep convolutional architecture for both the generator and discriminator, which improved the quality of generated images. CGAN introduced the concept of conditioning the generation process on additional information, such as class labels or other images. This allowed for more controlled image generation, which is particularly useful in image inpainting tasks.

Later works focused on stabilizing the training process, as it was often unstable and difficult to converge. The introduction of Wasserstein Generative Adversarial Network (WGAN) (Arjovsky et al., 2017) and Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) (Gulrajani et al., 2017) were important milestones in this regard. WGAN introduced a new loss function based on the Wasserstein distance, which provided smoother gradients and more stable training, while WGAN-GP added a gradient penalty term to further improve stability.

One pivotal development for image inpainting GANs were Context Encoders, introduced in 2016 (Pathak et al., 2016). Context encoders are a type of convolutional autoencoder specifically designed for image inpainting tasks. The architecture consists of an encoder, which compresses the input image with missing regions into a lower-dimensional latent representation, and a decoder, which reconstructs the image from this representation while filling in the missing areas. The model is trained using a combination of reconstruction loss, which measures the pixel-wise difference between the generated and ground truth images, and adversarial loss, which encourages the generator to produce more realistic images. This dual loss function helps the model learn both low-level details and high-level semantics, resulting in more convincing inpainted images. The model was able to reconstruct a fixed-size missing region, typically a square in the center of the image.

The Globally and Locally Consistent Image Completion (GLCIC) model, introduced in 2017 (Iizuka et al., 2017), was another influential work in the field of image inpainting using GANs. The main innovation of this model was the introduction of a dual discriminator architecture, which consists of a global discriminator and a local discriminator. The global discriminator evaluates the overall coherence of the generated image, ensuring that the inpainted region blends seamlessly with the surrounding context. The local discriminator, on the other hand, focuses on the details within the inpainted region, ensuring that textures and fine details are realistic. This dual approach allows the model to capture both global structure and local details, leading to more convincing inpainted results. The generator architecture is based on an encoder-decoder structure, with dilated convolutions to increase the receptive field without losing resolution. This allows the model to reconstruct arbitrarily shaped missing regions while maintaining high-quality details.

Another early influential model, DeepFill, was originally introduced in 2018 (J. Yu et al., 2018). A second version of the model was introduced in 2019 (J. Yu et al., 2019). Its architecture introduced two notable concepts: contextual attention (in DeepFill v1) and gated convolutions (in DeepFill v2).

Contextual attention allows the model to borrow features from distant areas of the image which are semantically similar to the missing region. This layer works by finding patches in the known regions of the image which are similar to the patches being generated. These patches' features are then used to guide the inpainting of the missing regions. This helps the model generate coherent and realistic content, as it better understands global context, and it can identify plausible patterns in other regions of the image. Figure 2.10 illustrates the architecture of DeepFill.

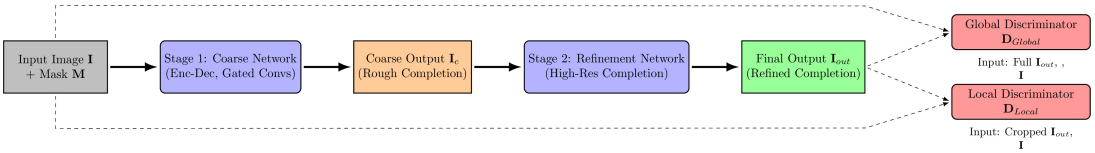

Figure 2.10: *DeepFill v1 architecture, featuring a two-stage coarse-to-fine generator and dual discriminators (global and local). The first stage generates a coarse prediction of the missing region, while the second stage refines this prediction to produce a more detailed and realistic output. The dual discriminators ensure both global coherence and local detail in the generated images. Adapted from (J. Yu et al., 2018).*

Gated convolutions deal with the limitations of standard convolutions regarding irregular masks. They feature a dynamic selection mechanism for each channel and spatial location, which can itself learn a gate that controls the flow of information between input and output for each pixel and channel. By using this mechanism, the model can dynamically determine the validity and importance of each input pixel. As a result, the generated images show better color consistency and inpainting quality. Figure 2.11 illustrates the architecture of DeepFill v2.

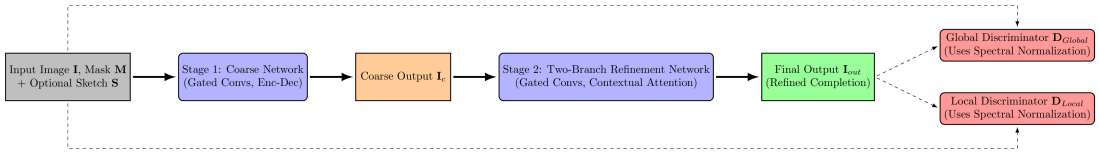

Figure 2.11: *DeepFill v2 architecture, featuring a two-branch refinement network with gated convolutions and dual discriminators (global and local). The two-branch design allows the model to separately process structural information and texture details, leading to more realistic inpainted images. The gated convolutions enable the model to effectively handle irregular masks by dynamically controlling the flow of information. Adapted from (J. Yu et al., 2019).*

PEPSI (M.-c. Sagong et al., 2019), introduced in 2019, is another notable model dedicated to image inpainting. It features a two-stage coarse-to-fine architecture, where the first stage generates a coarse prediction of the missing region, and the second stage refines this prediction to produce a more detailed and realistic output. The model also incorporates a spatially discounted reconstruction loss, which assigns higher weights to pixels near the boundary of the missing region, encouraging the model to focus on these areas during training. Furthermore, PEPSI employs a multi-scale discriminator, which evaluates the generated images at different scales, ensuring that both global structure and local details are realistic. In 2021, the authors introduced a lighter version of the model, Diet-PEPSI (Shin et al., 2021), which achieves similar performance with fewer parameters. Figure 2.12 illustrates the architecture of PEPSI.

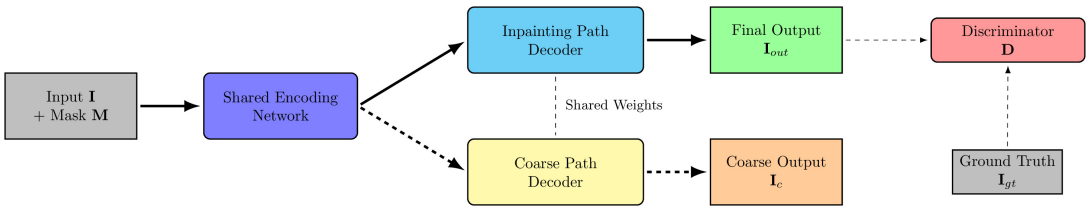

Figure 2.12: *PEPSI architecture, featuring a two-stage coarse-to-fine generator with shared weights between the coarse and refinement paths, and a multi-scale discriminator. The coarse path generates a rough prediction of the missing region, while the inpainting path refines this prediction to produce a more detailed and realistic output. The shared weights help maintain consistency between the two stages, and the multi-scale discriminator ensures that both global structure and local details are realistic. Adapted from (M.-c. Sagong et al., 2019).*

Conditional Modulation Generative Adversarial Network (CoModGAN) (Zhao et al., 2021), introduced in 2021, adapted a generative network for inpainting. This network, StyleGAN, had been introduced by Nvidia researchers in late 2018 (Karras et al., 2019). The authors of CoModGAN were able to modify the generator network to create new objects or scenes within large missing regions, instead of relying on the surrounding pixels.

CoModGAN achieves this by combining two different types of GANs: image-conditional GANs and unconditional modulated GANs. Image-conditional GANs take an image with a missing area and learn to fill it, while preserving existing content and ensuring consistency. On the other hand, unconditional modulated GANs generate images from

noise and use modulation techniques to control the style and features of the generated images. They can create diverse and high quality images from scratch. CoModGAN uses both an image conditional representation, which extracts features from the input image to understand its context and structure, and a stochastic noise representation, which creates a "style" vector from noise, allowing the network to generate new content. These representations are then combined into one, which is used to modulate the generator's convolutional layers. This way, the model can generate an image which conforms to the original image's context and structure, while also introducing new features, structures and textures, thus avoiding the typical blurriness found in typical GANs.

More recently, in 2024, Chen Y. et al. developed two new GAN-based models, Multi-scale Feature Module with Attention Module (MFMAM) (Y. Chen et al., 2024b) and Deep Neural Networks and Attention Mechanism (DNNAM) (Y. Chen et al., 2024a). MFMAM introduces a multi-frequency masked attention mechanism, which separates the input image into high and low frequency components. The high-frequency component captures fine details and textures, while the low-frequency component captures broader structures and color information. By processing these components separately, the model can better reconstruct both fine details and overall structure in the inpainted regions. The model also employs a multi-scale architecture, where features are extracted at different scales, allowing it to capture both local details and global context. This is particularly useful for inpainting tasks, as it helps maintain consistency with the surrounding areas of the image. DNNAM builds upon MFMAM by introducing a dual-nature network architecture. It features two parallel branches: one branch focuses on reconstructing structural information, while the other branch focuses on texture synthesis. The structural branch uses a series of convolutional layers to capture the overall layout and geometry of the image, while the texture branch employs a more complex architecture with attention mechanisms to synthesise realistic textures. The outputs from both branches are then fused together to create the final inpainted image.

Diffusion Models

Diffusion models have since become the state-of-the-art approach for image generation. Their ability to generate new, high quality content can be highly beneficial for certain applications.

One key work in this field is *Denoising Diffusion Probabilistic Models* (Ho et al., 2020). This model laid the groundwork for future diffusion models, demonstrating their potential for high-quality image generation. The authors showed that by using a Markov chain to gradually add noise to an image and then learning to reverse this process, it was possible to generate high-quality images from random noise.

Diffusion models work by turning noise into an image. During their training process, the diffusion model must turn the input images into noise, and then be able to iteratively revert it back into the original image. These models can be used for inpaint-

ing tasks, such as object removal, by conditioning the reverse diffusion process with the masked image. They are able to generate higher quality images, with diverse inpainted content and fewer artifacts. Furthermore, they can better capture complex distributions and intricate details.

One of the earlier diffusion models for image inpainting, LaMa (Suvorov et al., 2021), was introduced in 2021. This model was able to inpaint large missing regions with high quality results. The generator uses Fast Fourier Convolution (FFC)s (Chi et al., 2020) in the generator, which allow the model to capture content from other areas of the image, even if they are far away from the missing region.

Another pivotal model is RePaint (Lugmayr et al., 2022), introduced in 2022. This model uses a pre-trained diffusion model and adapts it for image inpainting tasks. The model uses an iterative process to fill in the missing regions, where each step refines the inpainted area. The model performs multiple reverse steps (denoising) to generate the missing pixels, but at each step, it resamples the known pixels back to the ground truth. This conditioning ensures that the generated content seamlessly blends with the existing image data.

Stable Diffusion Inpainting (Rombach et al., 2022) is one of the more widely adopted solutions. The diffusion process occurs at a lower dimentional latent space, making it more computationally efficient. The original model was developed for generating images from text prompts. However, the inpainting version was modified and fine-tuned to take masked images as input.

Architectural Concepts

Both GANs and diffusion models share some important architectural aspects, such as the use of CNNs, attention mechanisms, transformers and coarse-to-fine approaches.

CNNs are typically used as the backbone of generative models, as they allow the model to learn complex features in images. A commonly used CNN is U-Net, described in section 2.2.1. The use of skip connections allows it to better reconstruct fine details in the inpainted images.

While CNNs are great at extracting local features, they often lack the ability to find long-range dependecies within the images. Attention mechanisms allow the model to identify relevant patches from known regions of the image to fill in the missing areas. Contextual attention, one of the earlier attention mechanisms, featured in the DeepFill model, allow the network to explicitly search for similar patches from the known areas of an image and use them to reconstruct the missing regions (J. Yu et al., 2018). This search is conducted by calculating a similarity score between features from different areas of the image. The similarity score acts as a weight for the model to aggregate information. Self-attention, as used in transformers, allows each element of an image (pixels or patches) to interact with every other element (Vaswani et al., 2023). The model is able to capture global dependencies and relationships across the image.

Transformers are neural networks which process sequential data in parallel, mak-

ing them computationally more efficient than traditional neural networks. While they were originally developed for Natural Language Processing (NLP) tasks (Vaswani et al., 2023), they have been extensively used in CV tasks.

Images are separated into smaller patches, called tokens, which are then converted into numerical embeddings Dosovitskiy et al., 2021. As the network would not normally be able to identify the position of each token, a positional encoding is added. The self-attention mechanism is used to compute Query (Q), Key (K) and Value (V) vectors for each token. The interaction between Q and K results in attentions scores, which are normalised and used to create a weighted sum of V vectors, which is essentially a new contextualised representation of the tokens. Several attention mechanisms, or "heads", allow the model to focus on different features of the image. The outputs of the attention heads are combined into one single output for each token, which is then fed into a Feed-Forward Network (FFN) to further process the information. The original transformer model uses an encoder-decoder architecture. The encoder consists of several identical layers, each with multi-head self-attention and an FFN. It results in a new contextual representation of the input sequence. The decoder also consists of several identical layers, but each has three sub-layers: masked multi-head self-attention, cross-attention (also known as encoder-decoder attention) and a FFN. The decoder's self-attention mechanism is similar to the one used in the encoder, but it is masked to prevent attending to future tokens. The cross-attention mechanism uses queries from the previous decoder layer and keys/values from the encoder's output, allowing the decoder to take the input sequence into account. The FFN is the same as the one used in the encoder. Figure 2.13 illustrates the architecture of a vision transformer.

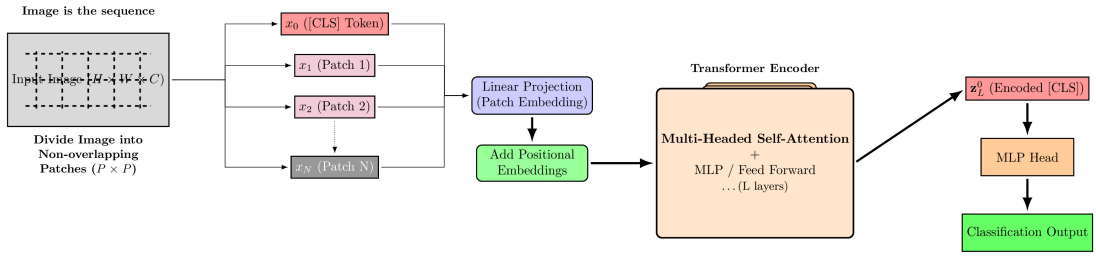

Figure 2.13: *Vision Transformer (ViT) architecture, illustrating the process of splitting an image into patches, embedding them, adding positional encodings, and passing them through multiple transformer layers consisting of multi-head self-attention and feed-forward networks. The final output is used for classification or other downstream tasks. Adapted from (Dosovitskiy et al., 2021).*

Due to the transformer's ability to model long-range dependencies in images, they have become highly attractive for inpainting tasks. They can either be used as a generative model's backbone, effectively replacing CNNs, or as components of a generative network in a hybrid approach (Esser et al., 2021). When used in a hybrid approach, a CNN is still used for initial local feature extraction, and a transformer is used at a later stage to identify relationships between different parts of the image. Some layers of the CNN can also be replaced with transformer blocks or modified by adding an attention

mechanism to improve the flow of information between known and masked regions of the image.

2.4 Datasets

Another important thing to consider when training a model is what it needs to be able to recognise. As such, finding a suitable dataset is paramount. In the research field, there are several prominent datasets which have contributed to better models being created. These datasets serve as a basis for comparison between different solutions.

For semantic segmentation, one of the most popular datasets is PASCAL Visual Object Classes (VOC) ((Everingham et al., 2010)). It was instrumental in the early development of semantic segmentation architectures, especially due to the introduction of the VOC 2012 segmentation challenge (Everingham et al., 2015). It contains 9,929 images, with 20 classes of objects.

Cityscapes (Cordts et al., 2016) is another important dataset for semantic segmentation, focusing on urban street scenes. It contains 5,000 high-resolution images with pixel-level annotations for 30 classes, including various object categories and stuff classes. The dataset is widely used for training and evaluating segmentation models in autonomous driving applications.

Scene UNderstanding (SUN) (Xiao et al., 2010; Xiao et al., 2016) is a large-scale dataset for scene understanding, containing over 130,000 images with pixel-level annotations for 397 object and stuff categories. It covers a wide range of indoor and outdoor scenes. A subset of this dataset, SUN RGB-D (Song et al., 2015), focuses on indoor scenes and includes depth information.

Places2 (Zhou et al., 2017) is a large-scale dataset for scene recognition, containing over 10 million images across 400+ scene categories. Similar to ImageNet, it is not specifically designed for segmentation tasks, but it has been used for pre-training segmentation models, especially for understanding scene context.

ADE20K (Zhou et al., 2018) is a comprehensive dataset for semantic segmentation, containing over 25,000 images with pixel-level annotations for 150 object categories. It covers a wide range of scenes from the SUN and Places2 datasets, including indoor and outdoor environments.

ImageNet (Deng et al., 2009) is a large-scale image dataset primarily used for image classification tasks. It contains over 14 million images across more than 20,000 categories. While it is not specifically designed for segmentation tasks, it has been used as a pre-training dataset for segmentation models, providing a rich source of visual features. A competition, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), has been held annually since 2010, driving significant advancements in computer vision (Russakovsky et al., 2015).

Common Objects in Context (COCO) (Lin et al., 2015) is one of the most influential and widely adopted datasets for instance segmentation and object detection. It also plays a significant role in semantic segmentation research and, since 2018, in panop-

tic segmentation research. It features a large number of everyday objects in complex, natural scenes, with pixel-wise instance masks for 80 object categories, and 91 "stuff" categories. The 2017 COCO-Stuff dataset (Caesar et al., 2018) contains approximately 118,000 images for training, 5,000 for validation and 41,000 for testing. In the training and validation splits, there are approximately 886,000 instances of objects.

Several datasets used for image segmentation models, such as ImageNet, Places2, COCO, have also been used for training inpainting models.

Typically, the images are combined with masks, resulting in "corrupted" images with missing areas. These masks can be random rectangles, irregular masks, or masks based on object segmentation.

The models are trained to infer the missing areas in order to achieve results similar to the original images. Some specialised datasets for image inpainting have also been released.

CelebA-HQ (Karras et al., 2018) is a high-quality version of the CelebA dataset (Z. Liu et al., 2015), containing 30,000 images of celebrity faces with high resolution. It has been used for training and evaluating image inpainting models, particularly for face inpainting tasks.

RORD (M.-C. Sagong et al., 2022) was released in 2021, as an effort to set a new benchmark for image inpainting models. It contains over 400,000 images for training and 50,000 images for validation. The images contain objects from COCO classes. Each image has an associated ground truth image, where objects are not present, and a binary mask denoting the objects to be removed. The authors showed significant improvements in model performance when trained in the RORD dataset, against the COCO and Places2 datasets with masks applied.

Syn4Removal (Jiang et al., 2025) is a large-scale object removal dataset, created by copying and pasting objects from instance segmentation datasets onto images. It provides these images both with objects, serving as input, and the original images as ground-truth. It contains 1 million training images, 10,000 validation images and 10,000 test images. The objects are taken from the COCO dataset, while the background images are taken from the ImageNet dataset. The authors showed that models trained on this dataset were able to generalise well to real-world images.

2.5 Training Methodologies

As evidenced by the constant improvement of network architectures, with the introduction of revolutionising concepts, it is clear that these are fundamental to achieve good results. However, just as important as the architecture itself are the training methodologies.

2.5.1 Loss Functions

For the training process, it is crucial to use an adequate loss function, which quantifies the difference between the desired output of the model and the target. The goal of the training process is to minimise this function as it learns how to reach the desired output. There are several loss functions which can be used in image segmentation.

For pixel-wise classification models, cross-entropy loss functions are commonly used (I. Goodfellow et al., 2016). The Binary Cross Entropy (BCE) loss function is used when it comes to binary segmentation. For each pixel, it compares the predicted probability of belonging to a certain class to the true value. In multi-class semantic segmentation, categorical cross-entropy is used instead. It compares the predicted probability distribution of each pixel over all classes to the true value. This loss function struggles dealing with class imbalance – when the number of pixels belonging to one class is much greater than the number of pixels belonging to other classes. The model may not be able to identify small objects in an image with other large objects, as the model prioritises identifying the class with the largest area. It also treats all pixels equally, regardless of their relationship or the object shape.

Cross-Entropy Loss:

$$\mathrm{CE} = -\sum_{i=1}^{N} y_i \log(p_i) \qquad \text{(Cross-Entropy Loss)}$$

where N is the number of classes, y_i is a binary indicator (0 or 1) if class label i is the correct classification for the pixel, and p_i is the predicted probability for class i.

Binary Cross-Entropy Loss:

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right] \qquad \text{(BCE Loss)}$$

where N is the number of pixels, y_i is the true label (0 or 1) for pixel i, and p_i is the predicted probability for pixel i.

Categorical Cross-Entropy Loss:

$$\mathrm{CCE} = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log(p_{ij}) \qquad \text{(Categorical Cross-Entropy Loss)}$$

where N is the number of pixels, C is the number of classes, y_{ij} is a binary indicator (0 or 1) if class label j is the correct classification for pixel i, and p_{ij} is the predicted probability for class j for pixel i.

The dice loss function (Dice, 1945) addresses the class imbalance problem by measuring the overlap between the predicted segmentation and the ground truth (Sudre et al., 2017). This way, it encourages the model to better segment smaller objects. However, it can lead to unstable training if the initial predictions are poor and there is no

overlap.

Dice Loss:

$$\text{Dice Loss} = 1 - \frac{2\sum_{i=1}^{N} p_i y_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} y_i + \epsilon} \quad \text{(Dice Loss)}$$

where N is the number of pixels, p_i is the predicted probability for pixel i, y_i is the true label (0 or 1) for pixel i, and ϵ is a small constant to prevent division by zero.

Another loss function which is similar to dice loss is the Jaccard Index loss function (Jaccard, 1901), also known as Intersection over Union (IoU) loss. While it shares the same advantages as the dice loss, it also suffers with the same problems. This function is often used for evaluating image segmentation models.

IoU Loss:

$$\text{IoU Loss} = 1 - \frac{\sum_{i=1}^{N} p_i y_i + \epsilon}{\sum_{i=1}^{N} p_i + \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} p_i y_i + \epsilon} \quad \text{(IoU Loss)}$$

where N is the number of pixels, p_i is the predicted probability for pixel i, y_i is the true label (0 or 1) for pixel i, and ϵ is a small constant to prevent division by zero.

The focal loss function aims to address extreme class imbalance and the problem of hard vs. easy examples by adding a modulating factor to the standard cross-entropy loss, down-weighting the loss contribution from well-classified examples and forcing the model to focus more on misclassified examples (Lin et al., 2018). The weighting factor for class imbalance and the focusing parameter must be carefully tuned for this loss function to work as intended.

Focal Loss:

$$\text{Focal Loss} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \quad \text{(Focal Loss)}$$

where p_t is the predicted probability for the true class, α_t is a weighting factor for class imbalance, and γ is the focusing parameter.

The Tversky loss function combines the dice loss and Jaccard loss, introducing two parameters that allow for control over the trade-off between false positives and false negatives (Salehi et al., 2017). If both parameters equal to 0.5, then the function equals the dice loss, and if they are equal to 1 it becomes the Jaccard loss. This function, just like in the case of the focal loss function, requires careful tuning of the parameters. If correctly tuned, it can provide fine-grained control over false positives/negatives, which is very valuable in specific applications, such as medical imaging.

Tversky Loss:

$$\text{Tversky Loss} = 1 - \frac{\sum_{i=1}^{N} p_i y_i + \epsilon}{\sum_{i=1}^{N} p_i y_i + \alpha \sum_{i=1}^{N} p_i (1 - y_i) + \beta \sum_{i=1}^{N} (1 - p_i) y_i + \epsilon} \quad \text{(Tversky Loss)}$$

where N is the number of pixels, p_i is the predicted probability for pixel i, y_i is the true label (0 or 1) for pixel i, α and β are parameters that control the trade-off between false positives and false negatives, and ϵ is a small constant to prevent division by zero.

It's common practice to combine loss functions to take advantage of the strengths of each. One such function combines the BCE and dice loss functions, and each is given a contributing factor which can be optimised.

Combined BCE and Dice Loss:

$$\text{Combined Loss} = \lambda \cdot \text{BCE} + (1 - \lambda) \cdot \text{Dice Loss} \qquad \text{(Combined Loss)}$$

where λ is a weighting factor that balances the contribution of each loss component.

2.5.2 Optimisers

To adjust the model's internal parameters, the loss value must be used by an optimiser, which tries to find the direction that will minimise the loss value in following predictions.

One of the better-known optimisers, upon which most other are built or compared, is the Stochastic Gradient Descent (SGD) optimiser (Robbins et al., 1951). It is simple and computationally efficient, as it calculates the gradient and updates parameters for a small random subset of the data at each step, instead of performing calculations over the entire dataset. It takes a step in the direction opposite to the gradient of the loss function. The size of the step is defined by the learning rate. Due to the stochastic nature of this optimiser, it can escape shallow local minima. However, it can be slow to converge, it is prone to oscillations and it is highly sensitive to the learning rate parameter. To mitigate some of these issues, a commonly used variant adds Momentum. This helps accelerating the optimiser in the relevant direction and dampens oscillations by adding a fraction of the previous update vector to the current update.

In the modern DL landscape, the Adaptive Moment Estimation (Adam) optimiser (Kingma et al., 2017) is one of the most popular and widely used optimisers. It automatically adjusts learning rates, often converges faster than SGD, particularly on complex models and datasets, and it handles sparse gradients well, which is a common problem in deep networks. The main drawback of Adam is that it can sometimes lead to suboptimal generalisation performance compared to SGD with momentum. To address this, several variants of Adam have been developed, such as Adam with Weight Decay (AdamW) (Loshchilov et al., 2019), which decouples weight decay from the gradient update, leading to better generalisation.

2.6 Evaluation Metrics

To understand how a model performs, it is important to evaluate it according to specific metrics.

When it comes to image segmentation, one of the most popular evaluation metrics is IoU/Jaccard Index (Jaccard, 1901). This metric measures overlap between predicted and ground truth masks. The Dice coefficient (Dice, 1945) is similar to IoU, and it is often used for assessing segmentation models for medical imaging.

For instance segmentation, the Average Precision (AP) metric is widely used. It is also used for evaluating object detection models. As for panoptic segmentation, the Panoptic Quality (PQ) metric (Kirillov et al., 2019b) combines segmentation quality and recognition quality metrics.

IoU/Jaccard Index (Jaccard, 1901): Intersection over Union, reports the overlap between the predicted segmentation and the ground truth.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{IoU}$$

Mean IoU (mIoU): The average IoU across all classes, providing a single performance measure for multi-class segmentation tasks.

$$\text{mIoU} = \frac{1}{C} \sum_{i=1}^{C} \text{IoU}_i \qquad \text{(mean Intersection over Union (mIoU))}$$

Where C is the number of classes and IoU_i is the IoU for class i.

Dice Coefficient (Dice, 1945): Similar to IoU, it measures the overlap between the predicted segmentation and the ground truth, but gives more weight to true positives.

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|} \tag{Dice}$$

Accuracy: Measures the overall correctness of the model's predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{Accuracy}$$

Precision: Evaluates the accuracy of the positive predictions made by the model.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{Precision}$$

Recall: Measures the ability of the model to find all the relevant cases (true positives).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{Recall}$$

F1 Score: The harmonic mean of precision and recall, providing a balance between the two.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{F1 Score}$$

Average Precision (AP): Measures the precision-recall trade-off for object detection and instance segmentation tasks. It is calculated by averaging the precision values at different recall levels.

$$\text{AP} = \int_0^1 p(r)dr \tag{AP}$$

Where $p(r)$ is the precision as a function of recall r.

Panoptic Quality (PQ) (Kirillov et al., 2019b): A comprehensive metric for panoptic segmentation that combines segmentation quality (IoU) and recognition quality (F1 Score).

$$\text{PQ} = \frac{\sum_{(p,g)\in TP} \text{IoU}(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \tag{PQ}$$

Where TP is the set of true positive matches between predicted segments p and ground truth segments g, FP is the number of false positives, and FN is the number of false negatives.

The evaluation of inpainting models requires specific metrics, which can be classified as either pixel-level or perceptual metrics. Pixel-level metrics include Mean Squared Error (MSE), PSNR and SSIM. Perceptual metrics include Fréchet Inception Distance (FID) and Learned Perceptual Image Patch Similarity (LPIPS).

Pixel-level metrics evaluate the model's performance by comparing the inpainted result to the ground-truth image pixel-by-pixel. They are sensitive to exact pixel matches. The MSE is the average of the squared differences between the pixel values of the ground-truth image and the inpainted image. While a lower MSE indicates a closer match between the inpainted image and the ground-truth image, this does not directly correlate with human perception: small pixel shifts or blurs can result in higher MSE values, while the image "looks good". PSNR measures the ratio between the maximum possible pixel value and the power of the reconstruction error (noise). It is expressed in decibels and higher values indicate better results. Just like MSE, it doesn't directly correlate with human perception. SSIM measures the perceived similarity between ground-truth and inpainted images, taking into account luminance, contrast and structure, which better correlates with visual perception. It is computed locally by sliding windows and averaging the results. The results can range from -1 to 1, where 1 indicates perfect similarity.

MSE: Mean Squared Error, measures the average squared difference between the pixel values of the original and reconstructed images. Lower values indicate better quality.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{MSE}$$

Where Y_i is the pixel value of the original image, $\hat{Y}_i$ is the pixel value of the reconstructed image, and n is the total number of pixels.

PSNR: Peak Signal-to-Noise Ratio, measures the quality of the reconstructed image compared to the original image. Higher values indicate better quality.

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{PSNR}$$

Where MAX_I is the maximum possible pixel value of the image (255 for 8-bit images) and MSE is the Mean Squared Error between the original and reconstructed images.

SSIM: Structural Similarity Index, measures the similarity between two images. Higher values indicate better quality.

$$\text{SSIM} = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)} \tag{SSIM}$$

Where μ_X and μ_Y are the average pixel values, σ_X^2 and σ_Y^2 are the variances, and σ_{XY} is the covariance of the two images. C_1 and C_2 are small constants to avoid instability.

Perceptual metrics aim to assess how the model's results align with visual perception.

The FID metric (Heusel et al., 2018) compares the distributions of a large set of generated images with a large set of ground-truth images. It uses an Inception-V3 network to extract features, which are then used to calculate the Fréchet distance (a measure of similarity between two distributions) between the feature sets. A lower score means that the distributions of generated and ground-truth images are more similar, implying higher quality results. A score of 0 indicates identical distributions.

LPIPS (Zhang et al., 2018) measures the perceptual similarity between two image patches (or full images). The images are fed into a CNN, trained on human judgments of image similarity. A lower score indicates that the images are more perceptually similar.

2.7 Related Work

Some of the most relevant works in the field of image inpainting are discussed in this section.

Some early models focused on handling free-form masks and irregular holes, as these are more applicable to real-world scenarios, as well as making sure the inpainted content is visually realistic.

GLCIC (Iizuka et al., 2017) was the first image inpainting model to allow for free-form masks. It uses two discriminators to make sure that the inpainted content is both locally and globally consistent.

Image Inpainting for Irregular Holes Using Partial Convolutions (G. Liu et al., 2018) introduced Partial Convolution (PConv), which uses partial convolutions that only consider valid pixels in the convolution operation. This approach allows the model to better handle irregular holes and produce more realistic inpainted results.

DeepFill v1 (J. Yu et al., 2018) introduced the Contextual Attention Module (CAM) to better capture long-range dependencies in the image. This attention mechanism allows the model to focus on relevant parts of the image when filling in missing regions, effectively solving the blurry results problem seen in previous models.

DeepFill v2 (J. Yu et al., 2019) built upon the previous version by introducing gated convolutions, which allow the model to learn dynamic feature selection. This results in improved inpainting quality, especially for complex structures and textures.

Recently, some models have shifted from using GANs to diffusion models for image inpainting, as they have shown to produce high-quality and diverse results. Notable examples include LaMa (Suvorov et al., 2021), Stable Diffusion Inpainting (Rombach et al., 2022) and RePaint (Lugmayr et al., 2022). However, these models are computationally expensive and require significant resources for training and inference. Furthermore, they are more prone to hallucination, as their priority is to synthesize realistic content, which may not always align with the original image context (Quan et al., 2024). For object removal tasks, where the goal is to seamlessly remove an object while preserving the surrounding context, GAN-based models are often preferred due to their ability to maintain context and produce coherent results.

There are some image inpainting pipelines that focus on object removal, such as SmartEraser (Jiang et al., 2025), but these often rely on user input to define the object to be removed. This can be done through manual selection or by providing a rough mask. Other solutions, such as PowerPaint (Zhuang et al., 2024), require text prompts to define the object to be removed. While this approach can yield good results, it requires user intervention and may not be suitable for fully automated applications.

While the models mentioned above have heavily contributed to today's state-of-the-art image inpainting techniques, they serve one specific purpose: image inpainting. For practical applications, such as automated object removal, image inpainting is just one step of the process. The object must first be identified and segmented from the image before it can be removed and inpainted. This requires a separate model for segmentation, which adds complexity to the pipeline. ModInPainTor, the pipeline developed in this project, addresses this by integrating segmentation and inpainting models. This integration allows for a more streamlined object removal process, where the segmentation model identifies the object to be removed, and the inpainting model fills in the resulting hole. Furthermore, it is possible to identify the classes of objects to be removed, allowing for more targeted object removal.

3 Research Methodology and Solution Design

In this Chapter, we detail the architecture of the object removal solution developed in this project. We discuss the research design, the architecture of the ModInPainTor pipeline, and the InPainTor model, which serves as a basis for this work.

3.1 Research Design

For this project, we are employing a Design Science Research (DSR) approach. DSR is a research paradigm that focuses on the creation and evaluation of artifacts designed to solve identified problems (Hevner et al., 2004). It is particularly well-suited for fields like information systems and computer science, where the development of practical solutions is a primary goal. The DSR methodology, as outlined by Hevner et al., predicts two main goals for research: the creation of an innovative artifact and the generation of new knowledge through the evaluation of that artifact. The authors propose seven guidelines, starting with the Artifact Creation guideline, which emphasizes the importance of designing and developing an artifact that effectively addresses a specific problem. This artifact can take various forms, including algorithms, models, methods, or systems. Furthermore, the methodology emphasizes the three cycles: relevance, rigor and design. The relevance cycle ensures that the research addresses a real-world problem, the rigor cycle connects the research to existing knowledge and theories, and the design cycle focuses on the iterative process of building and evaluating the artifact. The seven guidelines are the necessary criteria for assessing the quality of DSR.

While Hevner et al. provide a comprehensive framework for DSR, Peffers et al. offer a more detailed process model that breaks down the research into specific activities. This process model is widely adopted in DSR projects and provides a clear roadmap for researchers to follow.

The DSR methodology presented in (Peffers et al., 2007) outlines seven key activi-

ties:

- **Problem Identification and Motivation**: Define the specific business problem and justify the value of the solution.
- **Define the Objectives for a Solution**: Determine the artifact's functional requirements and desired performance goals.
- **Design and Development**: Create the artifact.
- **Demonstration**: Show how to use the artifact to solve the problem.
- **Evaluation**: Measure the artifact's performance against the solution objectives.
- **Communication**: Document the problem, the artifact, its utility, and its contribution to the scientific community.
- **Iteration**: The process is often iterative, cycling back to earlier steps based on evaluation results.

In this project, we tackle the problem of object removal from images in an industrial context. This involves developing a solution that integrates image segmentation and inpainting techniques to effectively remove unwanted objects while preserving the overall image quality. Such a solution is crucial for protecting employee privacy and ensuring that sensitive information is not inadvertently captured in images.

To address this problem, we will design and develop an artifact named ModInPainTor. This artifact will be a pipeline that combines a segmentation model to identify objects in images and an inpainting model to remove these objects seamlessly. The development process will involve selecting appropriate models, integrating them into a cohesive system, and optimizing their performance for the specific use case.

We will demonstrate the artifact by applying it to a set of images from an object removal dataset, showcasing its ability to remove specified objects while maintaining the integrity of the remaining image content. Furthermore, we will evaluate the artifact using both quantitative metrics, such as PSNR and SSIM, to assess the quality of the inpainted images, and qualitative analysis to evaluate the visual appeal of the results. Additionally, we will analyze how the accuracy of the segmentation model impacts the overall performance of the inpainting process.

Finally, we will document our findings and contributions in this project, providing insights into the design and effectiveness of the ModInPainTor pipeline. This documentation will serve as a resource for future research and development in the field of image processing and object removal.

3.2 ModInPainTor

ModInPainTor is an integration pipeline which combines three stages of image processing: segmentation, creation of the binary mask, and inpainting. The pipeline takes an image, or a set of images, as its input, along with the COCO class IDs referring to the object classes that must be removed from the images. The segmentation and generative

networks are instantiated and their respective weights are loaded. The images are processed by the segmentation model, which outputs an n-channel mask, where n equals to the amount of initially selected classes. This multi-channel mask is converted into a binary mask, as the generative model only needs to know which pixels need to be modified. The original images and their binary masks are processed by the generative model, which outputs the final, inpainted images.

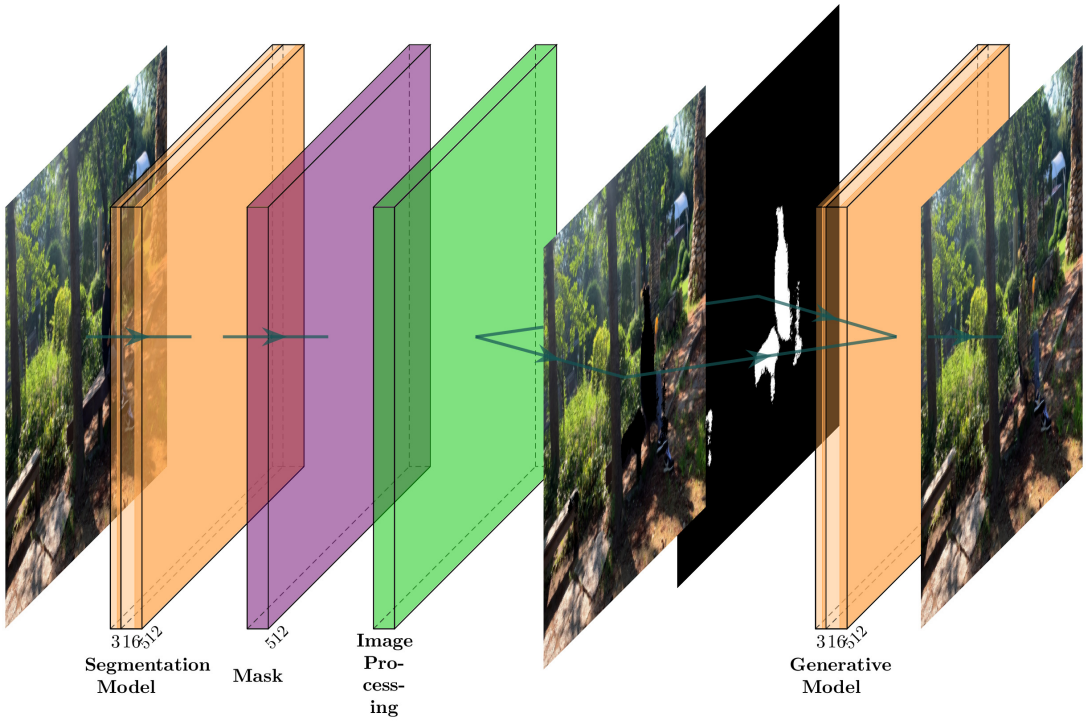

Figure 3.1: *ModInPainTor architecture*

The ENet model (Paszke et al., 2016) is specifically designed for real-time semantic segmentation. It uses an encoder-decoder architecture, where the encoder is responsible for extracting features from the input image, and the decoder is responsible for upsampling the feature maps to the original image size and producing the final segmentation masks. The model is designed to be efficient in terms of both memory and computation, making it suitable for real-time applications.

The ENet model is composed of a initial block, followed by five stages of bottleneck modules, and a final transposed convolutional layer. The initial block is composed of a convolutional layer with a kernel size of 3x3, stride of 2, and padding of 1, followed by a max pooling layer with a kernel size of 2x2 and stride of 2. This block reduces the spatial dimensions of the input image from 512x512 pixels to 256x256 pixels, while increasing the number of feature maps from 3 to 16. The bottleneck modules consist of a series of convolutional layers, batch normalization, and PReLU activation functions. The first stage contains 5 bottleneck modules, the second stage contains 8 bottleneck modules, the third stage contains 2 bottleneck modules, the fourth stage contains 2 bottleneck modules, and the fifth stage contains 1 bottleneck module. The first 3 stages

act as an encoder, while the remaining 2 stages decode the information into segmentation masks for each class. Each stage reduces the spatial dimensions of the input by half, while increasing the number of feature maps. The final transposed convolutional layer upsamples the feature maps back to the original image size of 512x512 pixels, and outputs n feature maps, where n equals to the number of classes to be segmented.

In comparison with the InPainTor's segmentation decoder, described in Section 3.3, the ENet model is deeper and more complex, with a larger number of parameters (see Table 4.3 in Section 4.2.2) and layers. This allows it to capture more intricate features and patterns in the input image, leading to better segmentation performance. The ENet architecture can be seen in Figure 3.2.

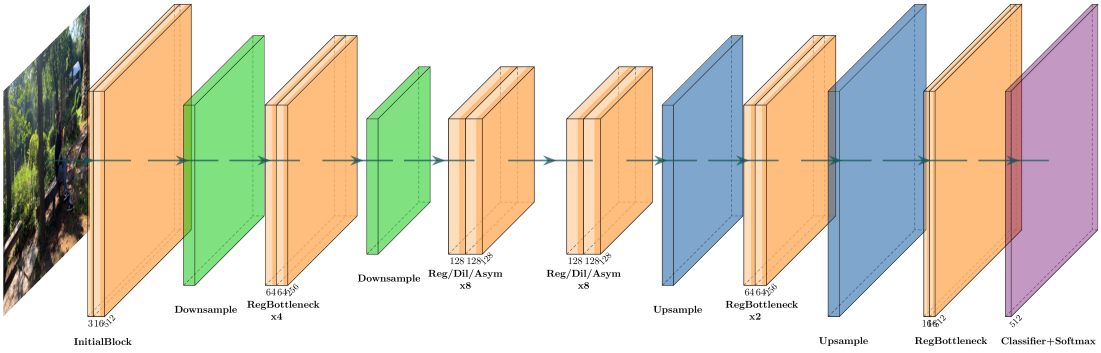

Figure 3.2: *ENet architecture*

For our purposes, we replaced the Softmax activation function in the final layer with a Sigmoid activation function, as we want to output independent probabilities for each class, rather than a probability distribution over all classes. This allows for the possibility of multiple classes being present in the same pixel, which is important for our application.

There are three types of DL based generative models: CNN, GANs and Stable Diffusion. As we discuss in Section 2.3, the best generative networks today fall into the category of GANs and Stable Diffusion.

While stable diffusion models are known to produce high-quality inpainting results, they prioritize the usage of random backgrounds for inpainted areas and are more prone to hallucination, whereas we want to maintain the original background as much as possible. Furthermore, diffusion models are known to be heavy on resources, which is a limitation we want to avoid. As such, we decided to use a GAN-based architecture.

From a selection of GAN-based architectures, some of which described in Section 2.3.2, we selected the PEPSI model (M.-c. Sagong et al., 2019) for its high performance in object removal tasks and its availability as open-source code (M.-c. Sagong, 2022). Furthermore, the authors of the PEPSI model, who also released the RORD dataset, performed an extensive comparison of their model against other state-of-the-art inpainting methods (as of 2022) (M.-C. Sagong et al., 2022), demonstrating an impres-

sive performance in terms of both quantitative metrics and qualitative results. Given the high quality of the results and the availability of the code, it was a natural choice for our object removal solution.

The PEPSI model (M.-c. Sagong et al., 2019) is a GAN-based architecture specifically designed for object removal tasks. It uses a coarse-to-fine approach, where a coarse network first generates a rough inpainting of the missing regions, and then a refinement network improves the details and quality of the inpainted areas. The model also incorporates a perceptual loss, which helps to preserve the overall structure and appearance of the image. The PEPSI model has been shown to outperform other state-of-the-art inpainting methods on several benchmark datasets, including RORD, Places2 and CelebA-HQ. One major advantage is its prompt availability as open-source code, which allows for easy implementation and experimentation.

The PEPSI generative model is composed of a generator and a discriminator. The generator uses an encoder-decoder architecture, with skip connections between the encoder and decoder layers. The encoder is used to extract features from the image, while the decoder presents two paths: a coarse and an inpainting (fine) path. During the training process, both paths are trained. First, a coarse inpainting result is generated, and then the fine path refines this result. The fine path contains a CAM, used for understanding the relation between background and hole regions. The CAM obtains information from the coarse results to understand where it should borrow information from in the image to complete the inpainted result. The generative architecture can be seen in Figure 3.3.

The Region Ensemble Discriminator (RED) is only used during training, and is discarded during inference. It uses a series of convolutional layers to extract features from the inpainted image and a set of FCLs to classify each region as real or fake. The discriminator is trained to distinguish between real and inpainted images, while the generator is trained to produce inpainted images that can "fool" the discriminator. The discriminator architecture can be seen in Figure 3.4.

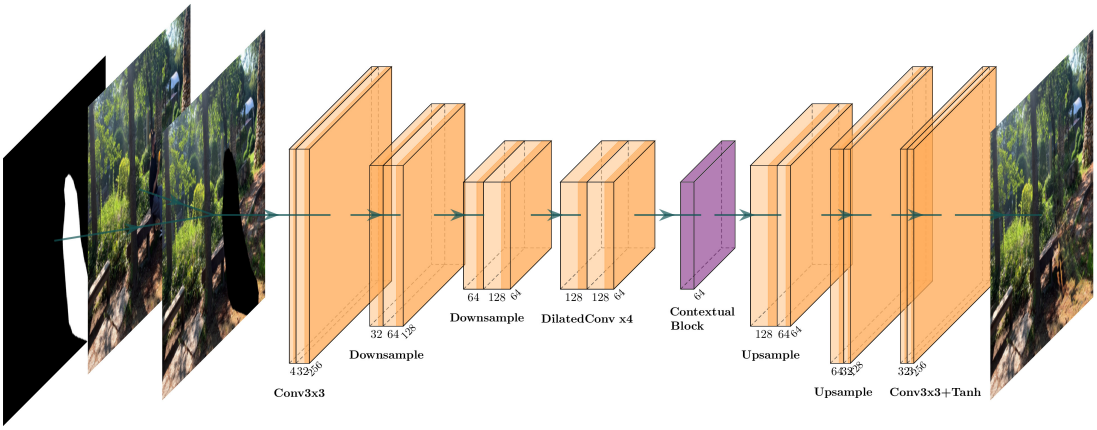

Figure 3.3: *PEPSI generative architecture*

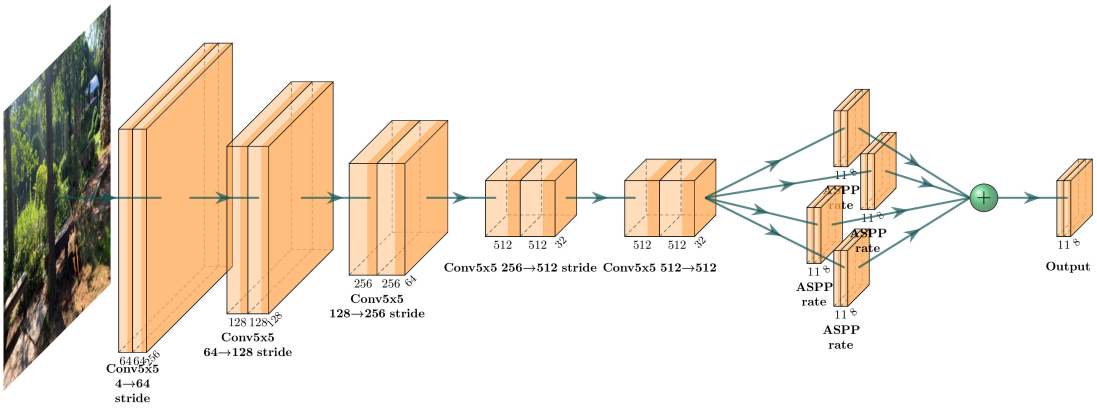

Figure 3.4: *PEPSI discriminator architecture*

3.3 InPainTor

InPainTor (Ribeiro, 2024) is an encoder-decoder architecture for image inpainting, more specifically for the task of object removal. This project tackles the problem of removing objects from images and filling in the missing areas with plausible content, while maintaining the overall structure and context of the image. The project was funded by FCT - Fundação para a Ciência e a Tecnologia, I.P..

The InPainTor network uses a shared encoder for feature extraction and two decoders: one for segmentation and one for inpainting. The segmentation decoder creates class masks from the feature maps obtained by the encoder, while the generative decoder removes objects from the original image taking into account the encoder's feature maps and the segmentation masks created by the segmentation decoder.

The InPainTor's architecture, due to the sharing of the encoder between segmentation and generative stages of the process, means that it is lightweight when compared to other solutions. The encoder uses convolution to create feature maps of smaller size, where each value condenses information from a broad area of the original image. Inversely, the decoders use these feature maps to create larger outputs - segmentation masks of 256x256 pixels and inpainted images of 512x512 pixels. While the segmentation decoder leverages the feature maps created by the encoder, not only from its last stage, but also from other stages where fewer feature maps are created but more information is available, the generative decoder also takes into account the masks and feature maps created by the segmentation decoder. This results in a resource-efficient model.

The shared encoder is composed of four separational convolutional blocks. The first one results in 32 feature maps of the same size as the input image. The remaining three halve the spatial dimensions of the input and double the number of feature maps. Each block is composed of two convolutional layers, followed by batch normalization and a ReLU activation function. The encoder takes an RGB image of 512x512 pixels as input, and outputs 256 feature maps of 64x64 pixels.

The segmentation decoder is composed of three separational transposed convolu-

tional blocks, each one doubling the spatial dimensions of the input and halving the number of feature maps. Each block is composed of a transposed convolutional layer, followed by batch normalization and a ReLU activation function. The first block takes the 256 feature maps of 64x64 pixels from the encoder as input, and the last block outputs 64 feature maps of 256x256 pixels. The 1x1 convolutional operation that follows outputs n feature maps of 256x256 pixels, where n equals to the number of classes to be segmented. Each upsampling block also takes as input the feature maps from the corresponding encoder block, through skip connections. This allows the decoder to capture both high-level and low-level features for better segmentation performance.

The generative decoder is composed of four upsampling blocks, each one doubling the spatial dimensions of the input and halving the number of feature maps. Each block is composed of a transposed convolutional layer, followed by batch normalization and a ReLU activation function. The first block takes the 256 feature maps of 64x64 pixels from the encoder as input, and outputs 256 feature maps of 64x64 pixels. The second block outputs 128 feature maps of 128x128 pixels, the third block outputs 64 feature maps of 256x256 pixels, and the fourth block outputs 3 feature maps of 512x512 pixels, which correspond to the RGB channels of the inpainted image. Each upsampling block also takes as input the feature maps from the corresponding encoder block, through skip connections. Additionally, the generative decoder takes as input the segmentation masks created by the segmentation decoder, which are concatenated to the feature maps from the encoder before being processed by the first upsampling block. The generative decoder also uses attention blocks, derived from feature maps created by the segmentation decoder. The feature maps generated by the attention blocks are then used to modulate the feature maps in the generative decoder. This allows the generative decoder to focus on the areas of the image that need to be inpainted, based on the segmentation masks.

The author notes some limitations about this architecture. In a first note, the model appears to struggle with large, diverse datasets such as COCO 2017. Data augmentation could aid in improving the model's performance, but it was yet to be implemented. The generative decoder's architecture also seems to struggle in producing good results, and the author states that it may be "too simplistic" for the problem. However, this model is light on resources and could prove to be a good choice where hardware constraints are found.

The author also provides some future directions for the development of the InPainTor project, namely in terms of improving segmentation performance and enhancing the generator's architecture.

Regarding the segmentation, one possible way forward would be to implement a more sofisticated architecture, such as ENet (Paszke et al., 2016) or Bilateral Segmentation Network (BiSeNet) C. Yu et al., 2018. ENet is a lightweight segmentation architecture comprising an encoder-decoder structure, with early downsampling and dilated convolutions to capture context while maintaining efficiency. BiSeNet uses a dual-path architecture, with a spatial path to preserve spatial information and a context

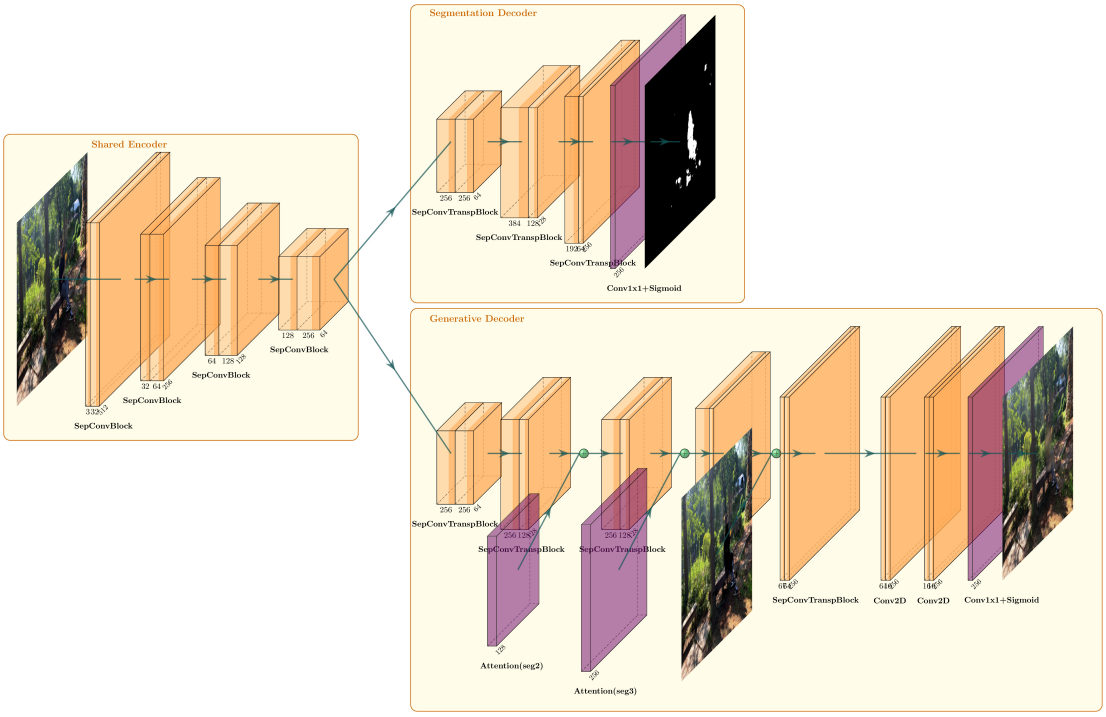

Figure 3.5: *InPainTor architecture*

path to capture high-level features. It also employs attention mechanisms to enhance feature fusion between the two paths. The BiSeNet authors compare their architecture against other segmentation architectures, including ENet. They report a 31.3% mIoU and 65.5% pixel accuracy on the COCO Stuff dataset when using ResNet101 as a backbone. On the Cityscapes dataset, they report a 68.7% mIoU when using Res18 as a backbone. On the same dataset, ENet reports a 51.3% mIoU, but higher IoU values in 8 out of 11 classes. Both architectures are designed to be efficient and could potentially improve the segmentation performance of InPainTor without significantly increasing its resource requirements.

As for the generative decoder, the number of parameters and layers could be tweaked as to maximize its performance. However, all of the components are tightly integrated, which means that larger modifications in one component could have severe impact on other components.

The InPainTor model had not yet been evaluated quantitatively, but the author provided some qualitative results, which show that the model is able to remove objects from images and fill in the missing areas with plausible content. The results are not perfect, but they are promising, especially considering the model's simplicity and resource efficiency.

In this project, InPainTor will be evaluated quantitatively and compared to other models. Furthermore, some of the author's suggestions for future work will be implemented in ModInPainTor and evaluated.

4
Implementation and Results

This section details the datasets and experimental setup used in our experiments, as well as the training and fine-tuning procedures for the segmentation and generative models. We also present the results obtained from our experiments, including quantitative metrics and qualitative analysis of the inpainting results. We conclude with a discussion of the findings and their implications for the field of object removal in images, as well as future directions for the project.

4.1 Experimental Setup

In this section, we detail the hardware and software environment used for our experiments, as well as the datasets employed.

4.1.1 Hardware

To train the segmentation and generative models, several machines were used. Their specifications can be seen in Table 4.1.

Table 4.1: *Hardware used for experiments*

Machine #	CPU	GPU	Memory	OS
1	AMD EPYC 7302	Nvidia A100 40GB	128GB	Ubuntu 20.04.6 LTS
2	Intel Xeon E5-2630	Nvidia Titan Xp 12GB	64GB	Ubuntu 22.04.4 LTS
3	AMD Ryzen 7 8845HS	Nvidia RTX4060M 8GB	32GB	Windows 11

Machine #2 was used to train the PEPSI generative model, while machine #3 was used to train the shared encoder and segmentation decoder on the InPainTor model for all 91 COCO classes.

All other tasks, including the optimisation of hyperparameters, testing and evaluation of individual models and end-to-end pipelines, were performed on machine #1

due to its significantly superior hardware performance.

4.1.2 Software

The main DL framework used was PyTorch, due to its flexibility, intuitive API, and strong support for GPU acceleration. Additional libraries and tools included:

- **NumPy** for numerical operations and array manipulation.
- **OpenCV** (**cv2**) for image processing and evaluation metrics.
- **Matplotlib** for plotting and visualization.
- **Optuna** for hyperparameter optimization.
- **tqdm** for progress bars during training and evaluation.
- **scikit-learn** for additional metrics and data processing utilities.

All experiments were managed and executed within isolated Miniconda environments to ensure reproducibility and dependency management.

To develop the code, we used Visual Studio Code, which allowed us to easily manage the different Python environments and seamlessly work across different machines. In remote machines, we used Visual Studio Code's remote development capabilities to connect to the machines via SSH and work directly on them.

When debugging certain features, we used Jupyter Notebooks, which allowed us to run code in an interactive manner and visualize outputs immediately.

When running commands on remote machines, we used tmux to create persistent terminal sessions. This allowed us to run long training processes even after disconnecting from the remote machine.

We used machine #1 for most tasks, including model evaluation, as it has the most powerful hardware. Machine #2 was used for training the PEPSI model, as machine #1 was being used for training sementation models. Machine #3 was used for training the InPainTor model's segmentation stage, as the other machines were being used to train other models.

4.1.3 Datasets

The selection of datasets for training, testing and evaluating models greatly impacts their performance. As such, it is important to choose adequate datasets for the intended tasks.

Both the InPainTor and ENet segmentation models were trained and evaluated on the COCO 2017 dataset. The COCO dataset contains over 200,000 images on the train set, and over 60,000 images on the validation set, with annotations for 91 classes of objects, enabling models to be more flexible. Furthermore, it is one of the most popular datasets for object detection and segmentation tasks, allowing for a broad comparison of models.

The train set was split in an 80/20 ratio to create a test set for final evaluation. The images have three channels, for RGB. For both InPainTor and ENet models, images were

resized to 512x512 pixels. The images were loaded in float32 format, in a range of [0, 1]. No normalization was applied to the images. The masks were created when loading the images, by projecting the polygons defined in the COCO annotations onto a blank mask of the same size as the image. Each class was assigned to a different channel in the mask tensor, resulting in a tensor of shape [number of classes, width, height]. The masks were also loaded in float32 format, in a range of [0, 1]. No data augmentation techniques were employed.

The InPainTor generative decoder and the PEPSI inpainting model were trained on the RORD dataset. As no optimization tasks were performed on the generative models, the models were trained on the test set and evaluated on the validation set.

For the InPainTor model, the original and ground truth images were resized to 512x512 pixels. The images were loaded in float32 format, in a range of [0, 1]. No normalization was applied to the images. As the InPainTor model uses the segmentation masks produced by the segmentation stage, the generative decoder was trained using these masks as input, alongside the original images. The dataset loader returns two arrays for each batch: one for the images, and the other for the class masks.

The PEPSI model was trained on images resized to 512x512 pixels. The images were loaded in float32 format, in a range of [0, 1]. The original and ground truth images were normalized to the range of [-1, 1]. The binary masks were loaded in float32 format, in a range of [0, 1]. The dataset loader return three arrays: one for the original images, another for the binary masks, and the other for the ground truth images.

For further evaluation purposes, a set of new ground truth images were created for the RORD dataset. These images were created by removing the persons from the original images using the labels provided in the dataset. We identify this dataset as the RPVD. The class IDs were reverse engineered, and the class pertaining to persons was identified as "F38". The masks were created by projecting the polygons defined in the RORD annotations onto a blank mask of the same size as the image. The areas defined in this new mask were then copied from the original ground truth images onto the original images, creating new ground truth images without persons. These images were used to evaluate the performance of the models in removing persons from images.

4.2 Model Training and Fine-Tuning

In this section, we detail the training and fine-tuning procedures for the segmentation and generative models used in our experiments. All models were trained from scratch to ensure a fair comparison of their performance.

4.2.1 Segmentation Models

The segmentation models output a tensor of shape [batch size, number of classes, width, height]. Each channel in the tensor corresponds to a class mask, with values

in a range of [0, 1], where 0 indicates that the pixel does not belong to the class, and 1 indicates that it does.

The training parameters for the InPainTor model were based on the original implementation, with some slight modifications to improve performance. We used the hyperparameters that were selected in the original implementation: a learning rate of 0.1, batch size of 8, the Adam optimizer and the BCE Loss. The scheduler was originally set to obtain the segmentation loss value after every 500 batches of images processed. If there was no loss improvement between two values, the learning rate would be updated by a factor of 0.25. However, we found the learning rate to be lowering too quickly, and decided to step the scheduler only after each epoch of training. Furthermore, instead of analysing the loss value, the scheduler was set to reduce the learning rate if the validation mIoU did not improve after 2 epochs. We also decided to reduce the learning rate by a factor of 0.1. We trained the shared encoder and segmentation decoder for 10 epochs, as had been specified by the original authors. We confirmed that there was no need to train for more epochs as by the 10th epoch there was no longer improvement in loss and mIoU values. While the author had originally trained the model to recognize 3 COCO classes, we decided to train it to recognize all 91 classes. To assess the impact of this decision on the model's segmentation performance, we decided to also train it for only 16 classes. The progression of the validation loss and mIoU for the model trained for all classes can be seen in Figure 4.1, and for the model trained for 16 classes can be seen in Figure 4.2.

The ENet model, obtained from (D. Silva, 2018), was trained on the COCO 2017 dataset, with all 91 classes, just like the InPainTor segmentation stage had been trained. In a first iteration of the model, we decided to use the hyperparameters that Paszke et al. describe in ENet's paper (Paszke et al., 2016). The learning rate was set to 5e-4, the batch size to 16 images, and we used the Adam optimizer. We used the ENet weighting function to calculate weights for each class according to their presence in the dataset. These weights were then used in a weighted loss function, where they were multiplied by the BCE Loss to aid the model in learning how to identify less common objects. To make sure the model achieved maximum convergence, we decided to train the model for 50 epochs. The training and validation loss and mIoU progression can be seen in Figure 4.3.

We allowed Optuna to optimize the hyperparameters of the ENet model, including the number of channels in the first convolutional layer, batch size, learning rate, and weight decay. To take advantage of the full capabilities of the hardware we were working on, we allowed Optuna to test the number of channels on the first convolutional layer in a range of 8 to 64, in steps of 2. Furthermore, the batch size was tuned in a range of 8 to 16, in steps of 2. The learning rate was tuned in a range of 1e-5 to 1e-3, and the weight decay between 1e-6 and 1e-3. The optimization process was guided by the validation mIoU metric. For each trial, the model was trained for a maximum of 5 epochs, with early stopping implemented to halt trials that did not show promising results. We chose AdamW for the optimizer. Optuna was allowed to run for 100

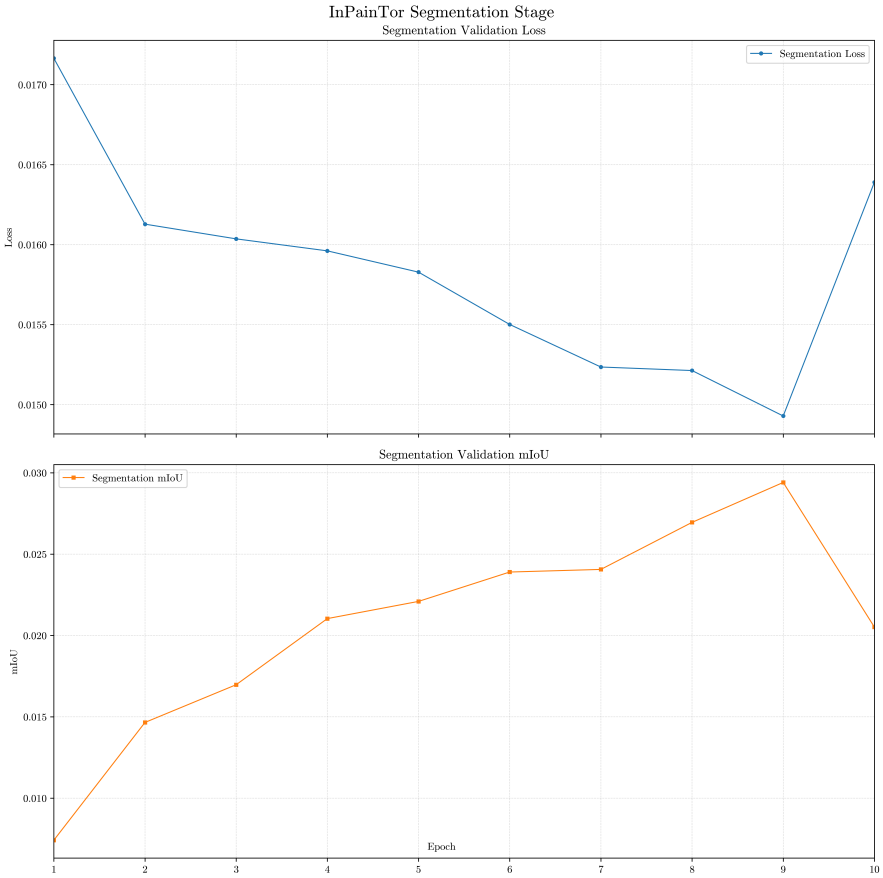

Figure 4.1: *InPainTor segmentation model validation loss and mIoU over 10 epochs.*

trials. The best hyperparameters were then used to train the ENet model for a full 50 epochs. The best results indicated the optimal value for the number of channels on the first convolutional layer to be 64, a batch size of 14 images, a learning rate of ~1.734e-4, and a weight decay of ~7.677e-6. The validation mIoU after 5 epochs of training was ~5.241%. The most important hyperparameter was found to be the number of channels on the first convolutional layer, at around an importance of 0.4, followed by the learning rate at around 0.16, the batch size at around 0.07 and the weight decay at around 0.05.

The hyperparameters are summarized in Table 4.2.

Table 4.2: *Segmentation Model Training Hyperparameters*

Description	Batch	Channels	LR	Opt.	Decay
InPainTor	8	16	0.1	Adam	None
ENet	10	16	5e-4	Adam	2e-4
ENet Tuned	14	64	~1.7343e-4	AdamW	~7.6770e-6

The ENet model was then trained using the finetuned hyperparameters, for 46 epochs. The model was originally planned to train for 50 epochs, but the process was halted at the 47th epoch as it was evident that there would be no more evolution in performance. There was a significant improvement in results, albeit at a high compu-

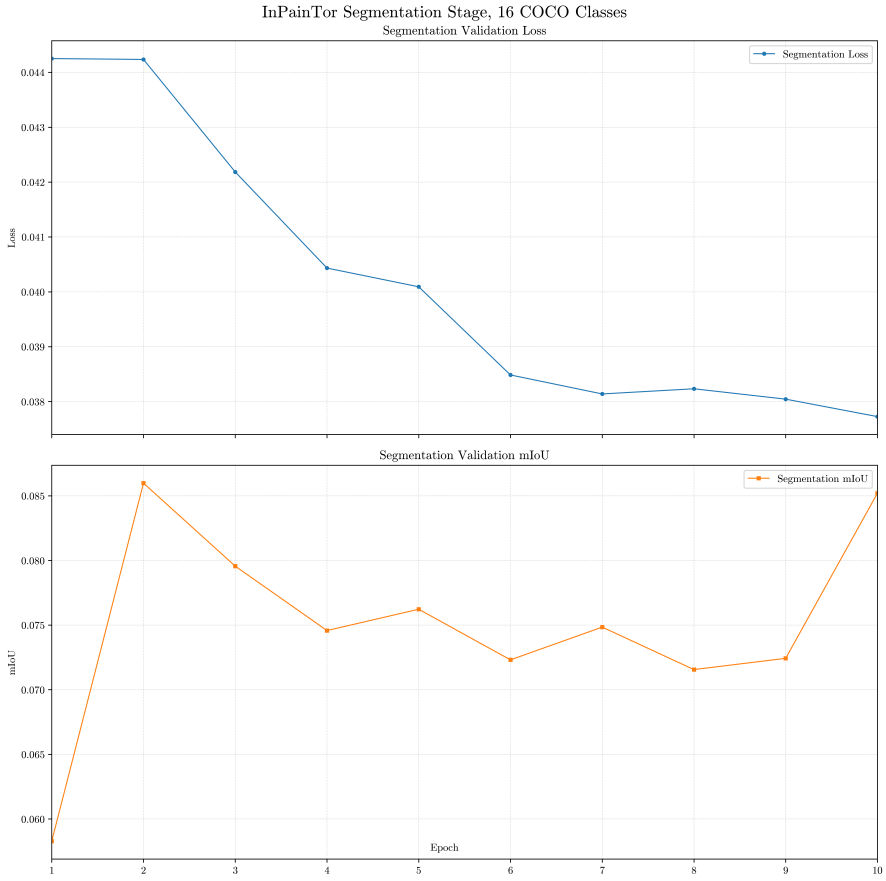

Figure 4.2: *InPainTor segmentation model for 16 classes validation loss and mIoU over 10 epochs.*

tational cost.

We also decided to train the model on 16 COCO classes, as to assess the impact of recognizing more classes. The training parameters for the tuned ENet model trained for 16 classes were the same as those used for the 91 class model. The only difference was that the class weights were recalculated to account for the different number of classes.

The validation loss and mIoU progression for the tuned ENet models can be seen in Figures 4.4 and 4.5.

4.2.2 Generative Models

The InPainTor generative stage is trained on the RORD dataset, after the segmentation stage, comprised of the shared encoder and segmentation decoder, is trained on the COCO dataset, as detailed in section 4.2.1. The encoder and segmentation decoder's weights are frozen, and the generative decoder is trained using the original images and the segmentation masks produced by the segmentation stage as input.The generative decoder is trained to reconstruct the original image without the objects identified in the binary mask. The MSE function is used. The model was trained for 10 epochs, with a batch size of 8 images. The Adam optimizer was used, with a learning rate of

Figure 4.3: *ENet original segmentation model validation loss and mIoU over 50 epochs.*

1e-1. The training loss progression can be seen in Figure 4.6.

The code for the PEPSI inpainting model was obtained from the RORD GitHub repository (M.-c. Sagong, 2022). Due to hardware constraints, the output image size was reduced to 256x256, and the batch size was reduced to 8 images. The model was trained for 450,000 iterations.

The PEPSI model uses the original images and binary masks as input. The original images are multiplied by the inverse of the binary masks, effectively blacking out the areas to be inpainted. The PEPSI generative model was trained alongside a discriminator, in an adversarial manner. After each batch of images was processed by the generator, the disriminator identified whether the inpainted images were real or fake. The loss function used for the generator was a combination of MSE loss and adversarial loss, while the discriminator used binary cross-entropy loss. The model was trained for 450,000 iterations, with a batch size of 8 images. The Adam optimizer was used, with a learning rate of 2e-4 for both the generator and discriminator. This learning rate was modified after 400,000 iterations, being reduced to 2e-5 for both models.

For both models, the output is a tensor of shape [batch size, 3, width, height], corresponding to the inpainted RGB images.

The ModInPainTor aggregates the segmentation model of choice with the PEPSI inpainting model. It takes a set of non-normalized images (i.e. in float32 format, in

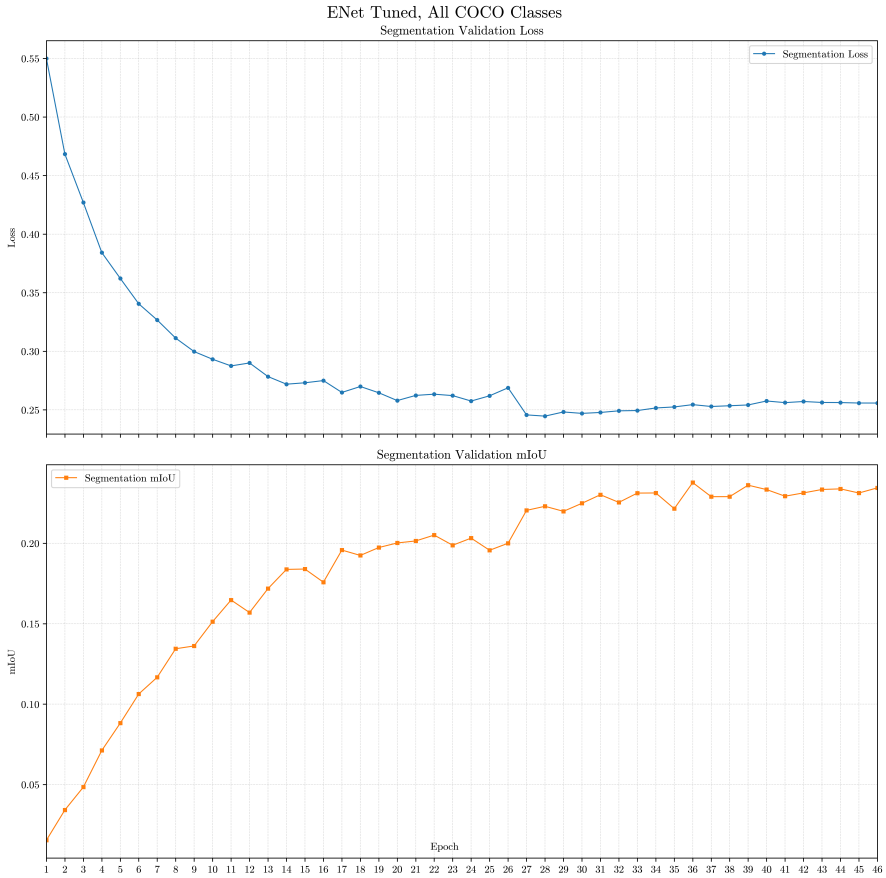

Figure 4.4: *ENet tuned segmentation model validation loss and mIoU over 46 epochs.*

the range [0, 1]) as input, and processes them through the segmentation model, which outputs a tensor of masks in the format [# of masks, # of classes, width, height]. The masks tensor is converted into binary masks. The original images must then be normalized to the range [-1, 1] to be compatible with the PEPSI model. The normalized images are multiplied by the inverted binary masks to blank out the areas to be inpainted. The blanked out images, along with their respective binary masks, are fed through PEPSI, which outputs the final images with the identified objects removed in a tensor of shape [batch size, 3, width, height]. These images are then denormalized back to the range [0, 1] to be comparable to the original images.

In Table 4.3, we can see a comparison between the models used in this project, in terms of their number of parameters and Giga Floating Point Operations Per Second (GFLOPS). The InPainTor model is the lightest, with only 0.52 million parameters and 19.54 GFLOPS. The ENet model, when tuned for 90 classes, has 5.53 million parameters and 45.02 GFLOPS, while the tuned version for 16 classes has slightly fewer parameters and GFLOPS. The PEPSI generator has 1.28 million parameters and 39.99 GFLOPS, while the discriminator is significantly larger, with 37.13 million parameters and 88.19 GFLOPS.

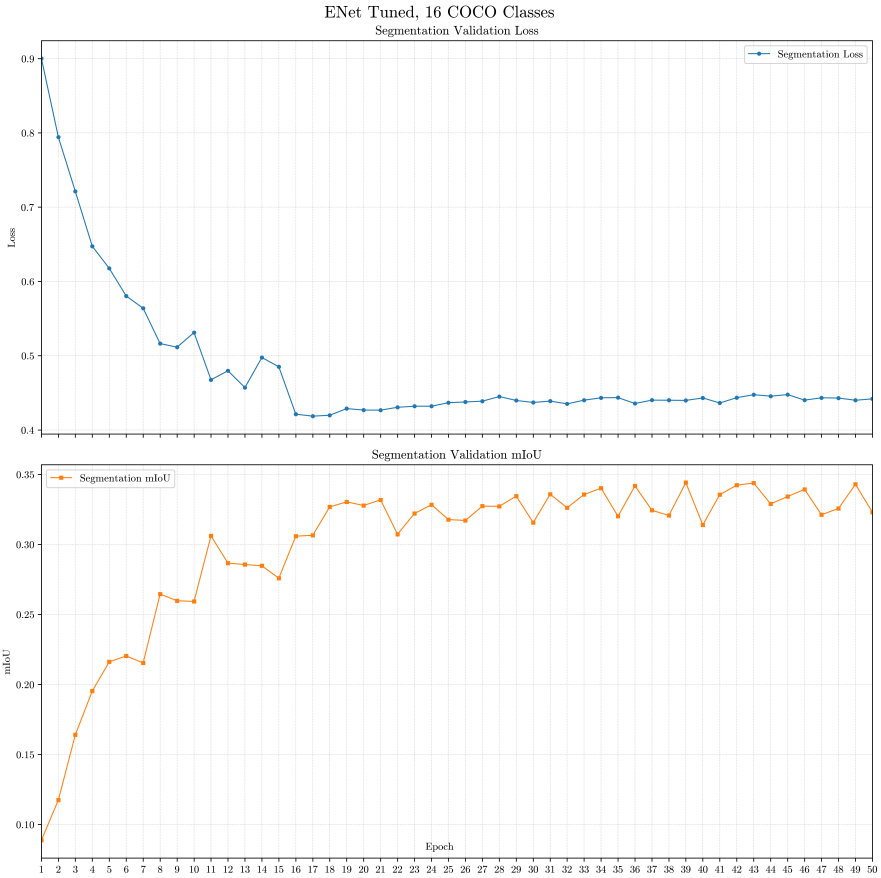

Figure 4.5: *ENet tuned segmentation model for 16 classes validation loss and mIoU over 50 epochs.*

4.3 Segmentation Analysis

In this Section, we detail the evaluation metrics and results for the segmentation models.

4.3.1 Evaluation Metrics and Results

The segmentation models were evaluated on the test set of the COCO 2017 dataset, obtained by subdividing the train set in an 80/20 ratio, according to the mIoU, Accuracy, Precision, Recall and F1 Score metrics. These metrics are obtained for each class mask, and derived from a confusion matrix. While the individual class evaluation metrics allow for some analysis on the performance of the model pertaining specific classes, their mean values are used to evaluate the performance of a model across all classes. The Accuracy value is not available for individual classes, as it is a global metric. The inference time was also measured, in Images Per Second (IPS).

The results of the segmentation models are presented in Table 4.4.

The new segmentation models show a significant improvement in performance when compared to the InPainTor solution. The original ENet model shows an increase in mIoU from 3.076% to 5.756% when trained for the 91 classes of the COCO dataset, al-

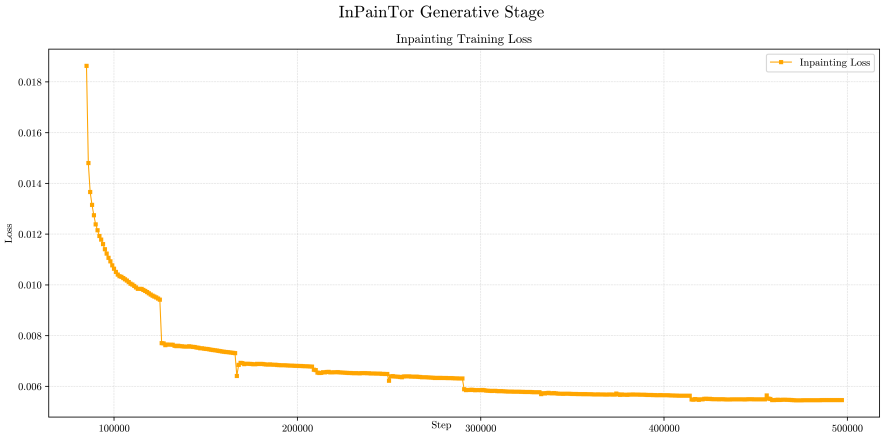

Figure 4.6: *InPainTor generative decoder training loss over 10 epochs.*

Table 4.3: *Model Parameter Analysis*

Description	Parameters (M)	GFLOPS
InPainTor	0.52	19.54
ENet Original, 91 cls.	0.36	5.50
ENet Tuned, 91 cls.	5.53	45.02
ENet Tuned, 16 cls.	5.48	33.70
PEPSI Generator	1.28	39.99
PEPSI Discriminator	37.13	88.19

though inference times are also increased from 4247.8721 IPS to 456.1780 IPS. The ENet model that was tuned specifically for the COCO dataset shows an incredibly large increase in performance to 18.678% mIoU. The tuned ENet model was able to process 165.6563 IPS, whereas the original ENet model was able to process 456.1780 IPS. While the ENet models are significantly slower than the InPainTor segmentation decoder, it is still able to process a large quantity of IPS. The tuned ENet model trained for only 16 classes shows an even larger increase in performance, reaching 25.515% mIoU. The inference times are comparable to the tuned model for 91 classes, resulting in 164.3777 IPS.

4.3.2 Qualitative Analysis

Some qualitative results of the segmentation models are presented and analyzed in this subsection. We divide this section into two parts: results on the 91 classes and results on the 16 classes.

Segmentation Results for 91 Classes

Some scenarios can be particularly challenging, particularly due to lighting conditions and occlusions. In these scenarios, the InPainTor model and the original ENet model heavily struggle with the segmentation of objects. Figures 4.7, 4.8 and 4.9 show an example of an image where two persons indoors cover a large area. In front of them is a table with dishes on top of it. The InPainTor model struggles with correctly identifying

Table 4.4: *Segmentation Models' Evaluation Metrics*

Description	mIoU	Accuracy	Precision	Recall	F1 Score	IPS
InPainTor, 91 classes	3.076%	36.939%	8.424%	7.715%	6.876%	**4247.8721**
InPainTor, 16 classes	7.712%	81.744%	10.010%	11.851%	11.467%	3960.2812
ENet Original, 91 classes	5.756%	55.554%	15.908%	12.922%	14.970%	456.1780
ENet Tuned, 91 classes	18.678%	50.498%	25.708%	43.166%	29.775%	165.6563
ENet Tuned, 16 classes	**25.515%**	**82.183%**	**32.681%**	**54.777%**	**37.807%**	164.3777

the persons, and only some small patches are correctly identified. Some patches of the table are misclassified as belonging to the person class. The original ENet model is able to identify a relatively large portion of the persons, but also misclassifies parts of the table as belonging to the person class. Furthermore, it is not able to identify that there is a table or dishes in the image. The tuned ENet model for 91 classes is able to correctly identify most of the persons, as well as part of the table and some dishes. It is not a perfect segmentation mask, but a significant improvement over the original model.

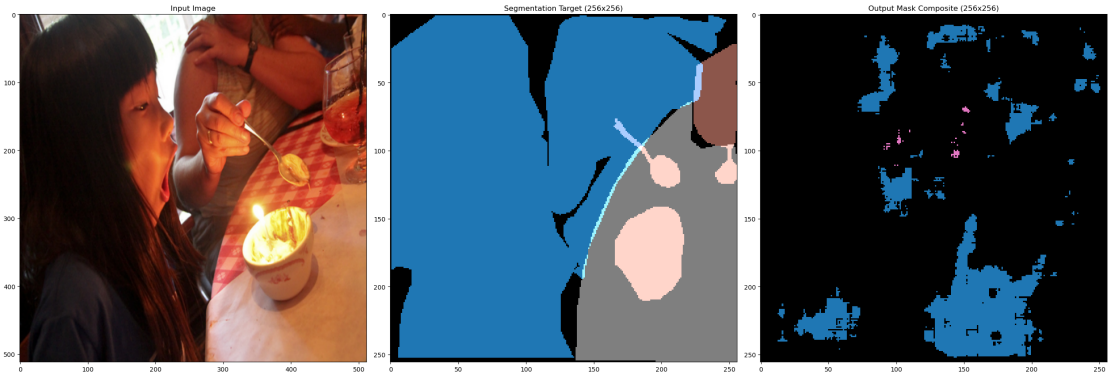

Figure 4.7: *Example of segmentation results from the InPainTor model. Two persons indoors cover a large portion of the image. A table stands in front of them, with dishes on top. From left to right: original image, ground truth mask, mask from segmentation model.*

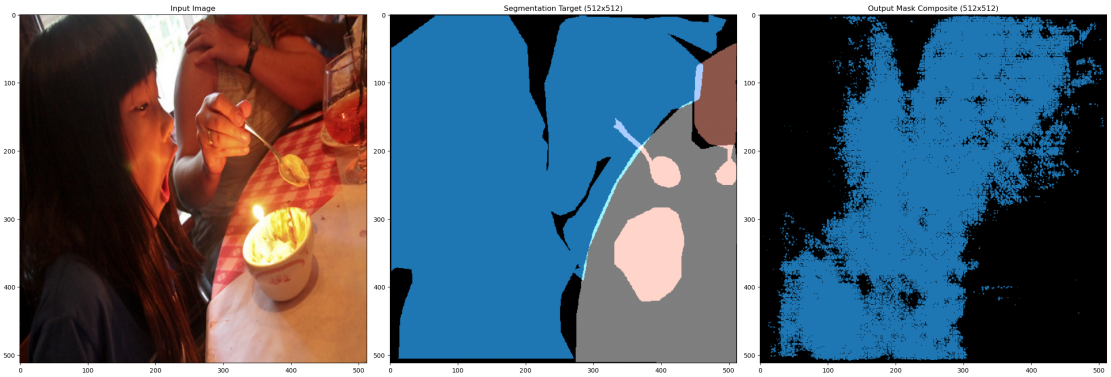

Figure 4.8: *Example of segmentation results from the original ENet model. Two persons indoors cover a large portion of the image. A table stands in front of them, with dishes on top. From left to right: original image, ground truth mask, mask from segmentation model.*

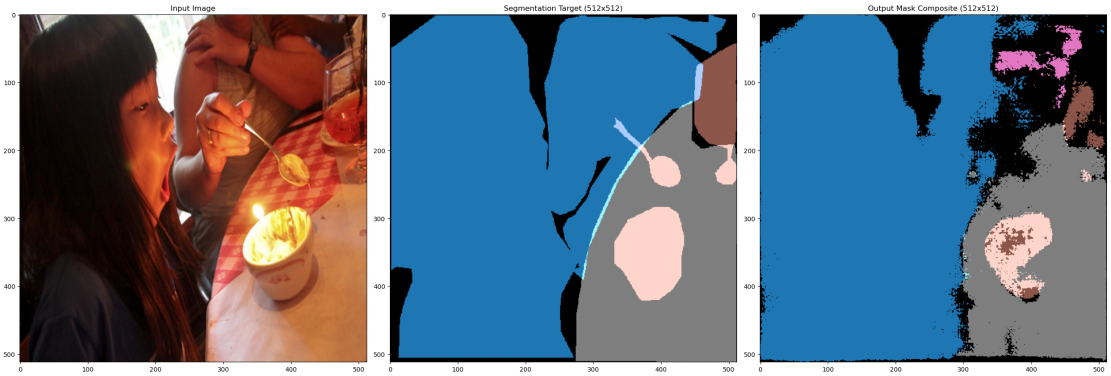

Figure 4.9: *Example of segmentation results from the tuned ENet model. Two persons indoors cover a large portion of the image. A table stands in front of them, with dishes on top. From left to right: original image, ground truth mask, mask from segmentation model.*

In classes less represented in the training dataset, the InPainTor model and the original ENet segmentation model struggle with the identification of objects. Figures 4.10, 4.11 and 4.12 show an example where the original image contains a group of elephants in a natural setting. The InPainTor model classifies some patches where the elephants stand as belonging to the person class. This indicates a bias towards the person class, as it is more prominent in the dataset. The original ENet model is not able to identify any object in the image. The tuned ENet model for 91 classes does a good job at identifying the elephants.

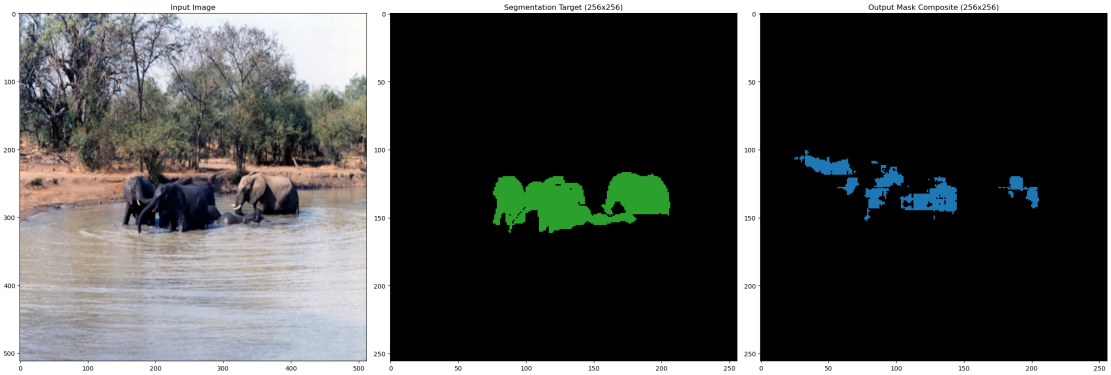

Figure 4.10: *Example of segmentation results from the InPainTor model. A group of elephants is present in a natural setting. From left to right: original image, ground truth mask, mask from segmentation model.*

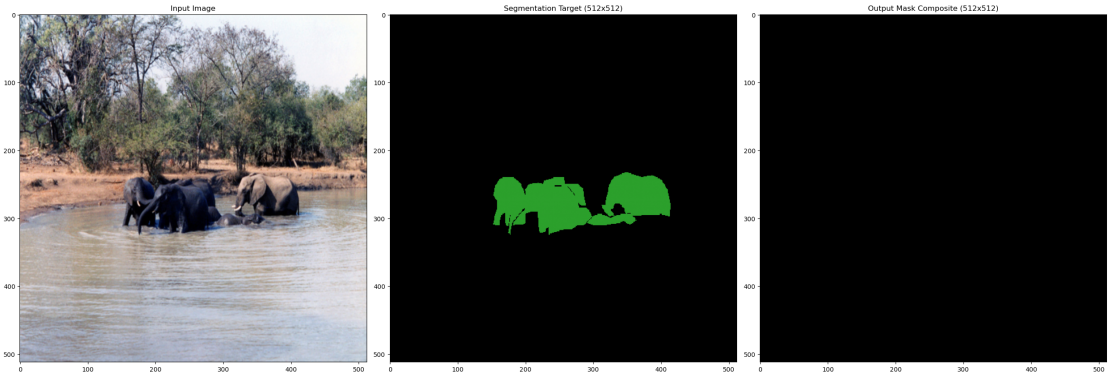

Figure 4.11: *Example of segmentation results from the original ENet model. A group of elephants is present in a natural setting. From left to right: original image, ground truth mask, mask from segmentation model.*

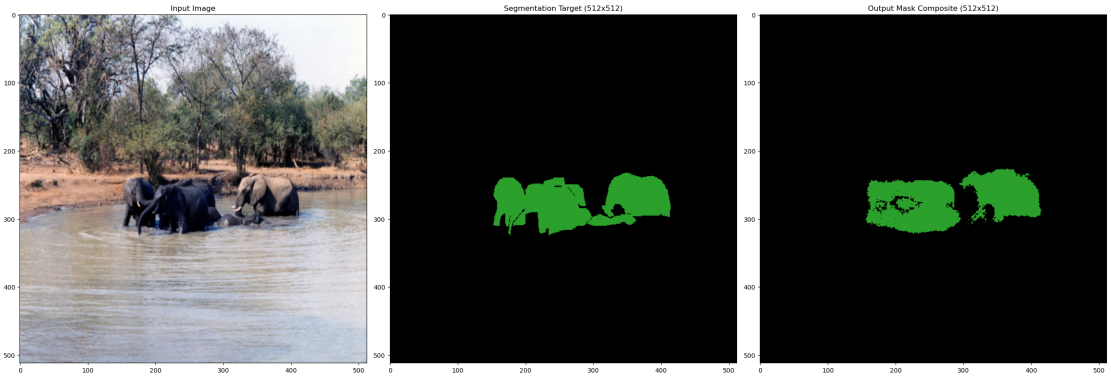

Figure 4.12: *Example of segmentation results from the tuned ENet model. A group of elephants is present in a natural setting. From left to right: original image, ground truth mask, mask from segmentation model.*

The next example, depicted in Figures 4.13, 4.14 and 4.15, shows another less represented class: pizza. The InPainTor model is only able to detect a small area of the border of the pizza on the lower right side of the image. Furthermore, it identifies some patches of the table cloth as belonging to the person class. The original ENet model is able to identify some patches of the pizza, while the tuned ENet model for 91 classes correctly segments a large area of the pizza.

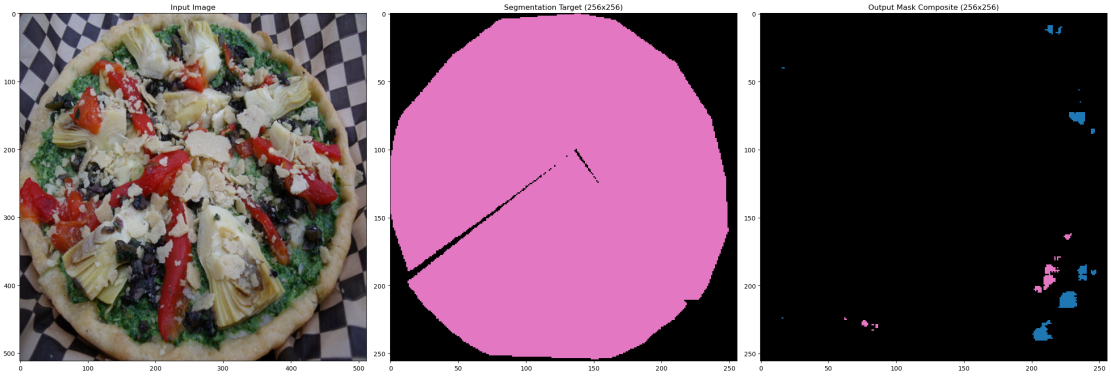

Figure 4.13: *Example of segmentation results from the InPainTor model. A pizza is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

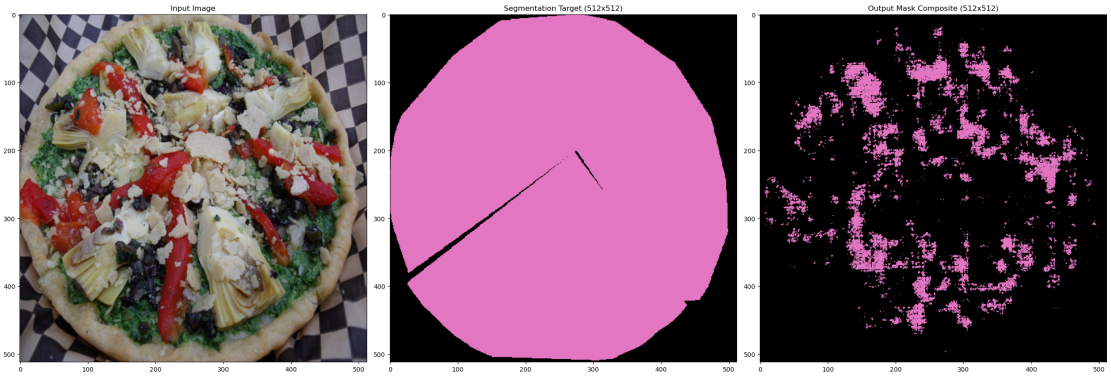

Figure 4.14: *Example of segmentation results from the original ENet model. A pizza is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

Figure 4.15: *Example of segmentation results from the tuned ENet model. A pizza is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

When it comes to persons, the models fare better. The person class is one of the most represented in the COCO dataset. The example in Figures 4.16, 4.17 and 4.18 shows a person in a suit. The InPainTor model can identify the head and the hand of the person. The original ENet model is able to identify a larger area of the person, but the mask is not very consistent. The tuned ENet model does a much better job at segmenting both the person and the tie. In the lower right area of the image, some patches of the background are misclassified as belonging to the person class. In the original image, we can see that the background is very dark, and of a similar color to the suit, which may have led to this misclassification. The area of the person's shoulder is also not fully identified. Overall, the tuned ENet model shows a significant improvement over the other two models.

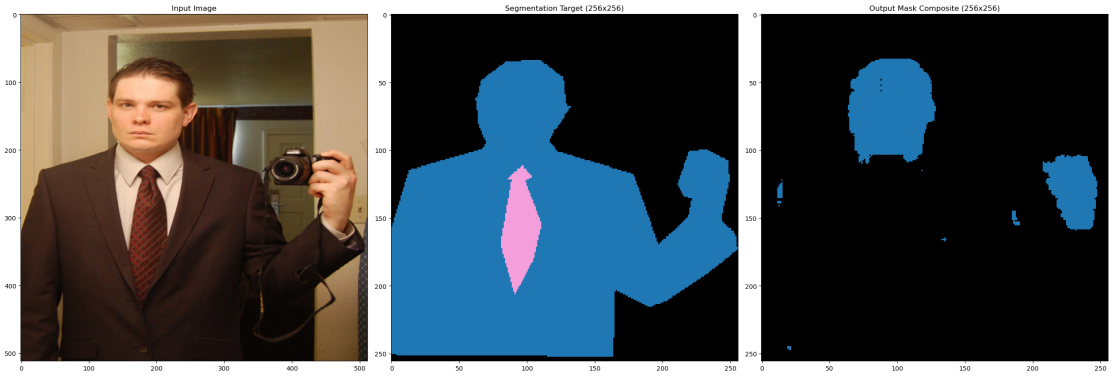

Figure 4.16: *Example of segmentation results from the InPainTor model. The image shows a person in a suit. From left to right: original image, ground truth mask, mask from segmentation model.*

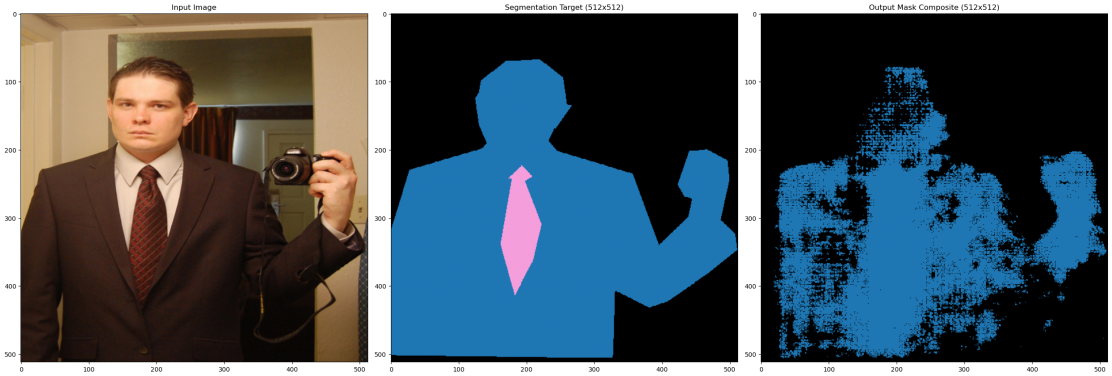

Figure 4.17: *Example of segmentation results from the original ENet model. The image shows a person in a suit. From left to right: original image, ground truth mask, mask from segmentation model.*

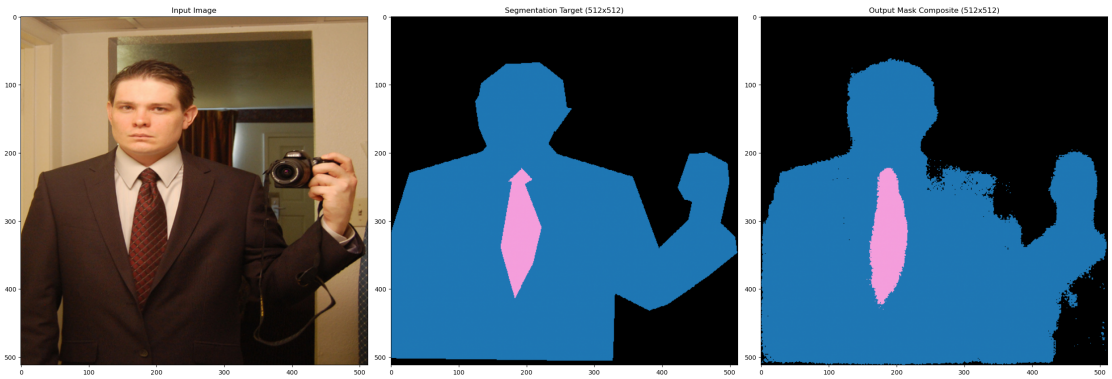

Figure 4.18: *Example of segmentation results from the tuned ENet model. The image shows a person in a suit. From left to right: original image, ground truth mask, mask from segmentation model.*

Some other larger objects are easier to be segmented by the models. Figures 4.19, 4.20 and 4.21 show a train covering a large area of the original image. The InPainTor model only identifies a few small patches of the train. The original ENet model is able to identify a much larger area of the train, although it is not very consistent. The tuned ENet model for 91 classes correctly segments most of the train, including finer details.

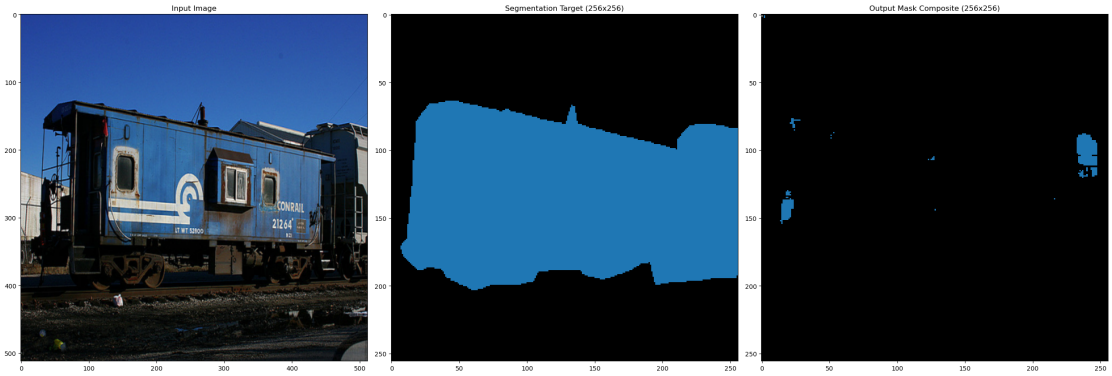

Figure 4.19: *Example of segmentation results from the InPainTor model. A train is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

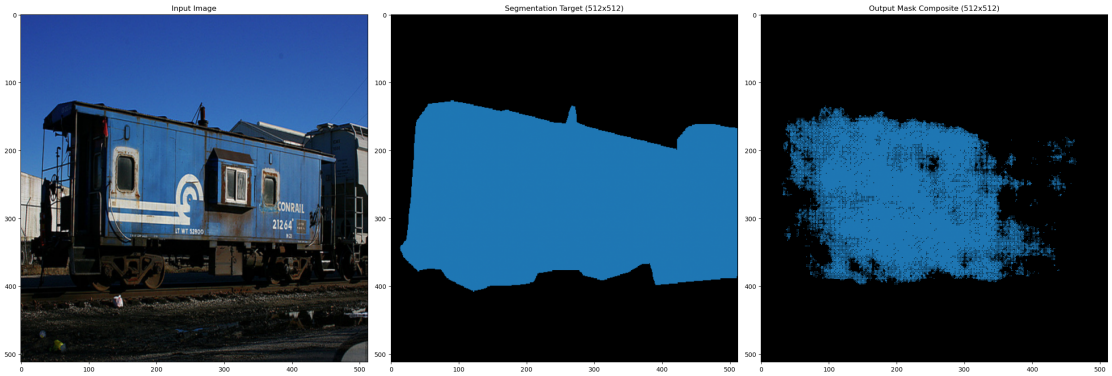

Figure 4.20: *Example of segmentation results from the original ENet model. A train is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

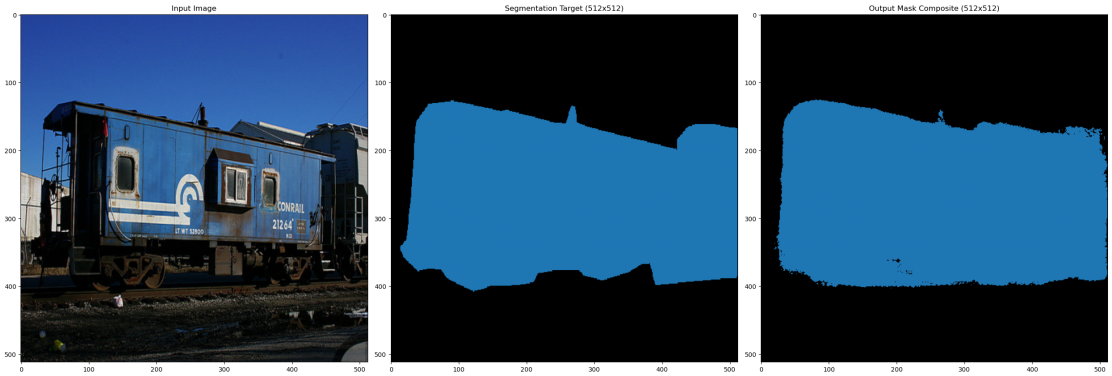

Figure 4.21: *Example of segmentation results from the tuned ENet model. A train is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

In complex scenarios, we see a repetition of the behaviour seen in other examples. Figures 4.22, 4.23 and 4.24 show a complex scenario: an airport scene, with multiple persons, vehicles, two airplanes and other objects. The InPainTor model is only able to identify some small random patches of the image. The original ENet model is only able to identify a portion of an airplane. The tuned ENet model correctly segments a large area of the airplane and vehicles, but it is not able to segment the persons, which

represent a small area of the image. It also misclassifies a patch at the bottom of the image. Still, it is a great improvement over the other models. The results can be seen in Figures 4.22, 4.23 and 4.24.

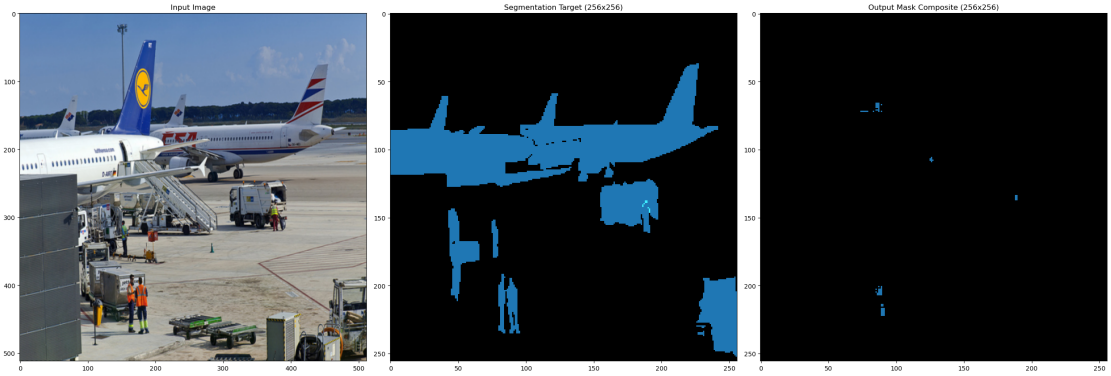

Figure 4.22: *Example of segmentation results from the InPainTor model. A complex scenario in an airport, with an airplane, vehicles and some persons. From left to right: original image, ground truth mask, mask from segmentation model.*

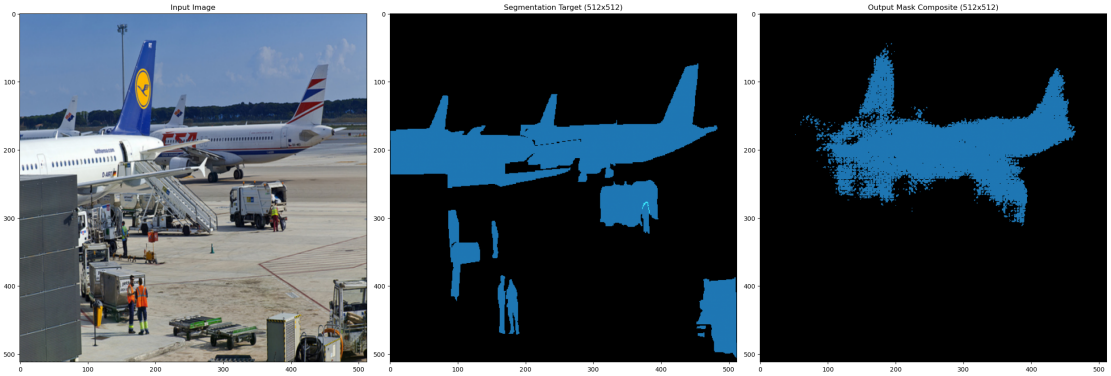

Figure 4.23: *Example of segmentation results from the original ENet model. A complex scenario in an airport, with an airplane, vehicles and some persons. From left to right: original image, ground truth mask, mask from segmentation model.*

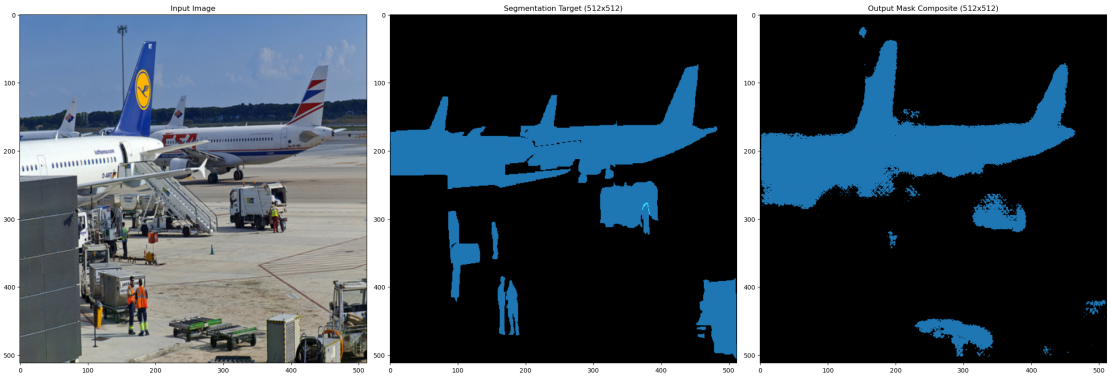

Figure 4.24: *Example of segmentation results from the tuned ENet model. A complex scenario in an airport, with an airplane, vehicles and some persons. From left to right: original image, ground truth mask, mask from segmentation model.*

Segmentation Results for 16 Classes

The segmentation results for the models trained for 16 classes are presented in this subsection. The classes selected were the first 16 on the COCO dataset: person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird and cat.

In the first example, depicted in Figures 4.25, 4.26, 4.27 and 4.28, we see a 3D generated image with one person laying in an hospital bed, and another sitting on the edge of the bed. Here, the person sitting covers a large area of the image. The InPainTor model for 16 classes is only able to identify some patches of the persons. The InPainTor model for 91 classes shows a remarkable improvement in performance over the InPainTor model for 16 classes, being able to segment most of the persons, although the mask still lacks several patches. The tuned ENet model for 16 classes is able to correctly segment most of the person sitting, although the mask is not consistent. Within the mask, some patches are not classified as belonging to an object. Furthermore, it segments the person under the bedsheets. As the tuned ENet model for 91 classes is able to identify other objects in the image, it struggles with identifying the person laying in the bed.

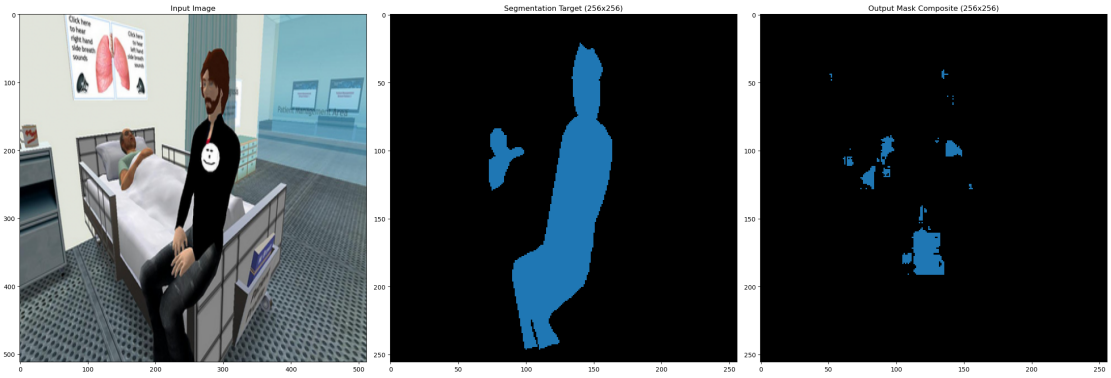

Figure 4.25: *Example of segmentation results from the InPainTor model trained for 16 classes. A person is laying in an hospital bed, and another is sitting on the edge of the bed. From left to right: original image, ground truth mask, mask from segmentation model.*

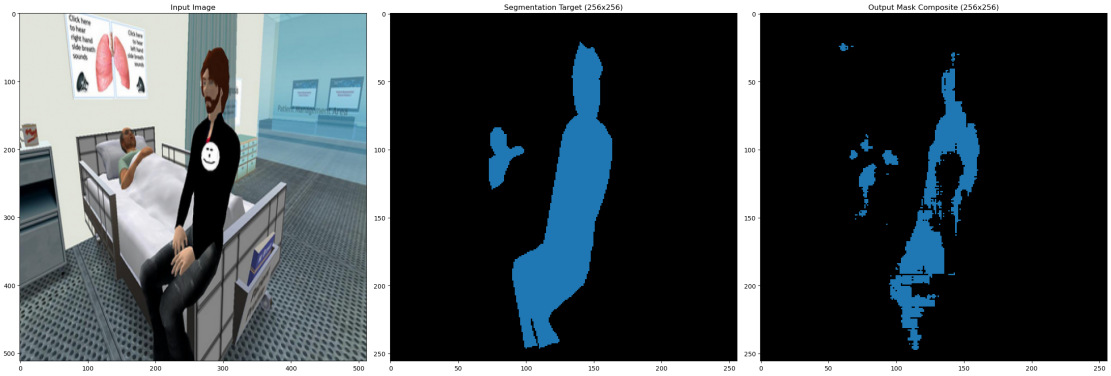

Figure 4.26: *Example of segmentation results from the InPainTor model trained for 91 classes. A person is laying in an hospital bed, and another is sitting on the edge of the bed. From left to right: original image, ground truth mask, mask from segmentation model.*

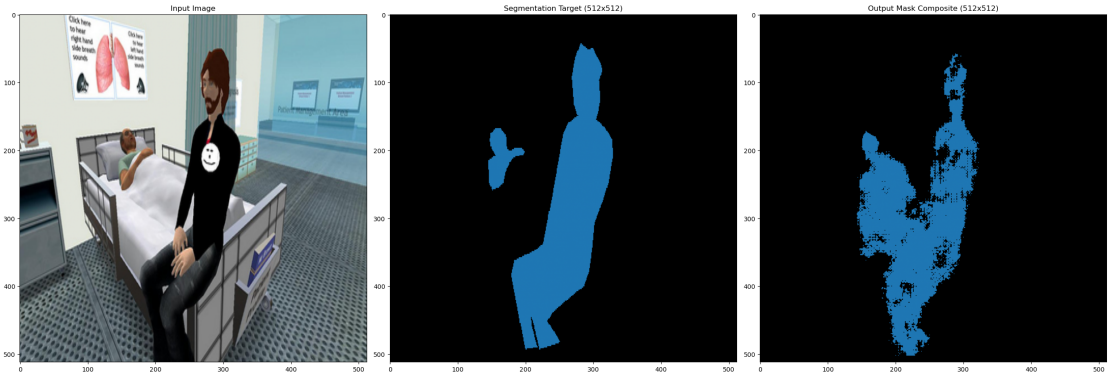

Figure 4.27: *Example of segmentation results from the ENet model trained for 16 classes. A person is laying in an hospital bed, and another is sitting on the edge of the bed. From left to right: original image, ground truth mask, mask from segmentation model.*

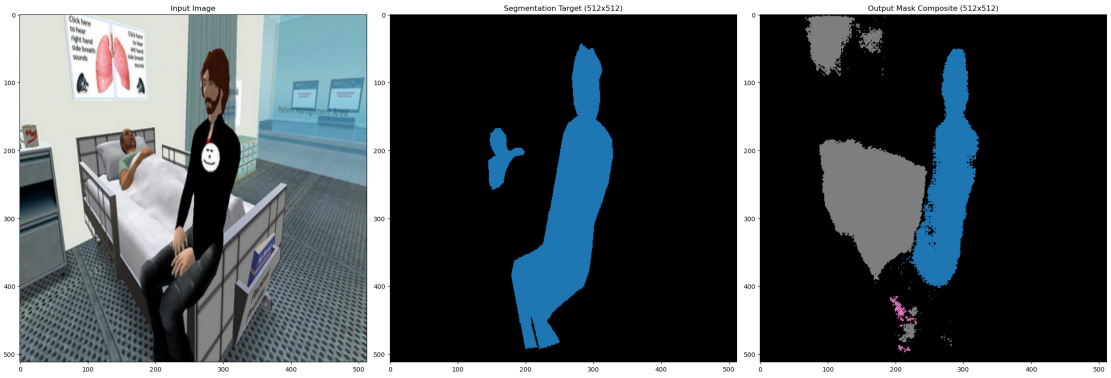

Figure 4.28: *Example of segmentation results from the InPainTor model trained for 91 classes. A person is laying in an hospital bed, and another is sitting on the edge of the bed. From left to right: original image, ground truth mask, mask from segmentation model.*

In the next example, depicted in Figures 4.29, 4.30, 4.31 and 4.32, we see a person standing behind a large couch. This is a challenging indoors scenario, with a light source from the window in the back of the room, creating complex shadows and highlights. Furthermore, the person represents a small portion of the image. The InPainTor model for 16 classes is not only able to identify the person and misclassifies a small patch of the background as belonging to the person class. The InPainTor model for 91 classes shows similar performance, misclassifying several patches of the floor as belonging to the person class. The tuned ENet model for 16 classes is able to segment the face of the person, while the tuned ENet model for 91 classes struggles to identify the person. However, it can correctly segment the couch.

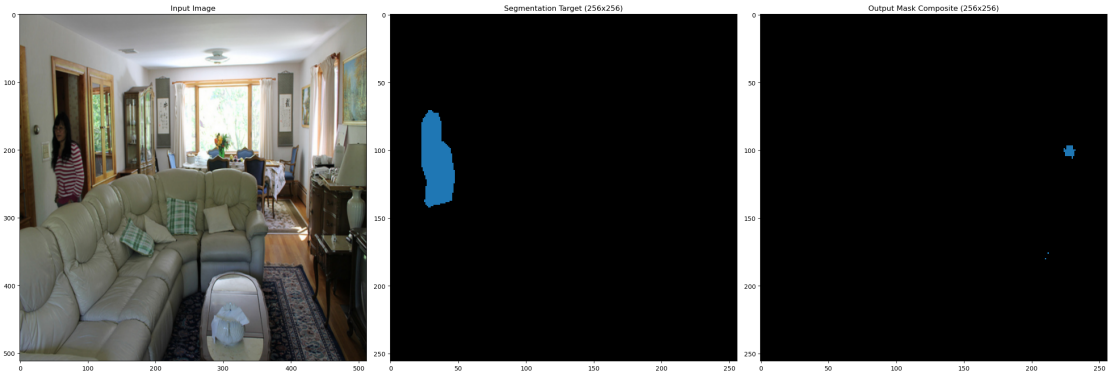

Figure 4.29: *Example of segmentation results from the InPainTor model trained for 16 classes. A person is standing behind a large couch. From left to right: original image, ground truth mask, mask from segmentation model.*

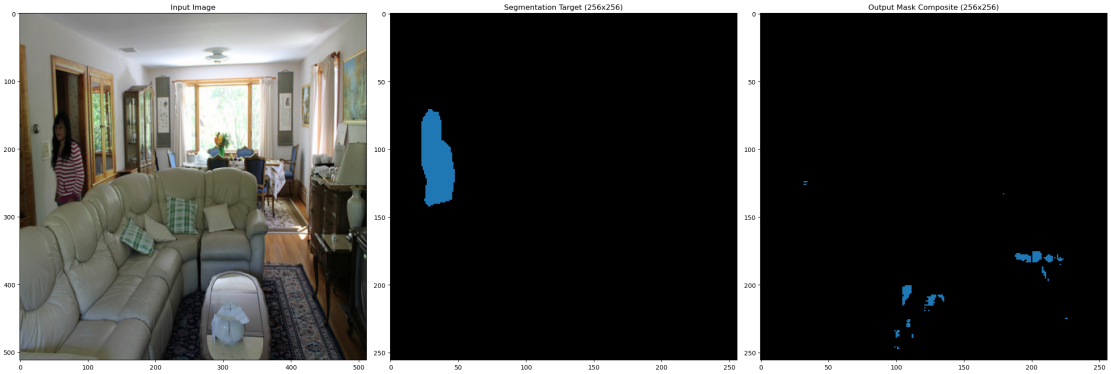

Figure 4.30: *Example of segmentation results from the InPainTor model trained for 91 classes. A person is standing behind a large couch. From left to right: original image, ground truth mask, mask from segmentation model.*

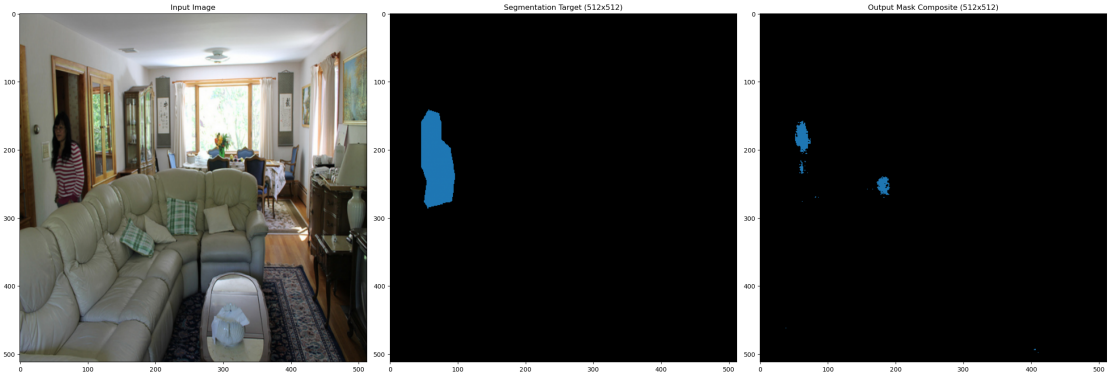

Figure 4.31: *Example of segmentation results from the ENet model trained for 16 classes. A person is standing behind a large couch. From left to right: original image, ground truth mask, mask from segmentation model.*

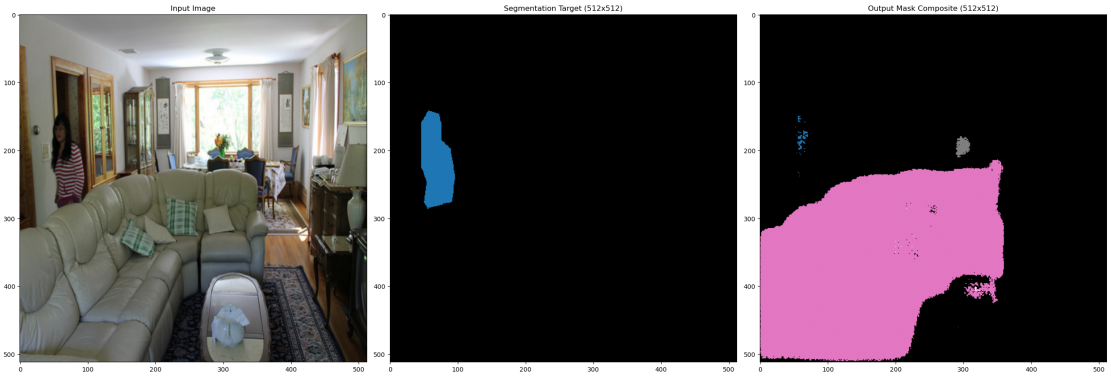

Figure 4.32: *Example of segmentation results from the InPainTor model trained for 91 classes. A person is standing behind a large couch. From left to right: original image, ground truth mask, mask from segmentation model.*

The third example is depicted in Figures 4.33, 4.34, 4.35 and 4.36. The original image was taken outdoors, and shows a food truck and several people standing around it. The InPainTor model for 16 classes is only able to identify some patches of the persons, and misclassifies several patches across the image. The InPainTor model for 91 classes performs similarly. The tuned ENet model for 16 classes does a stellar job, correctly segmenting most of the persons, as well as the food truck. The tuned ENet model for 91 classes performs similarly, and identifies the umbrella on the right side of the image.

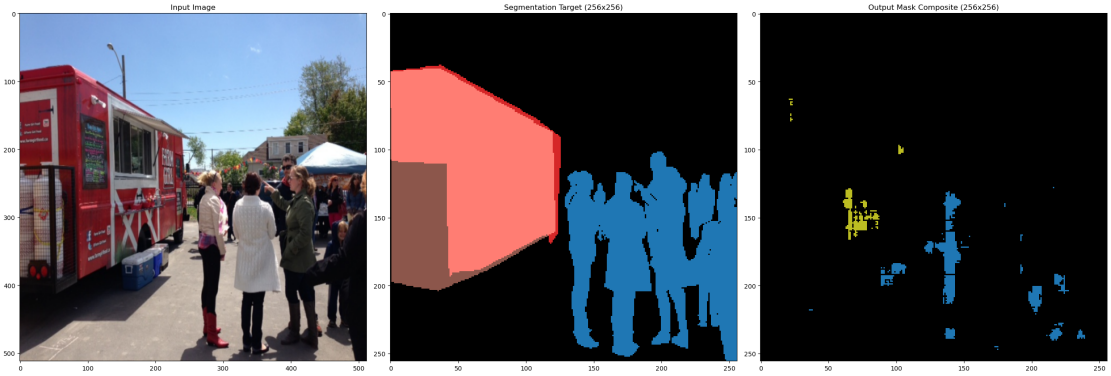

Figure 4.33: *Example of segmentation results from the InPainTor model trained for 16 classes. A food truck and several persons standing around it are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

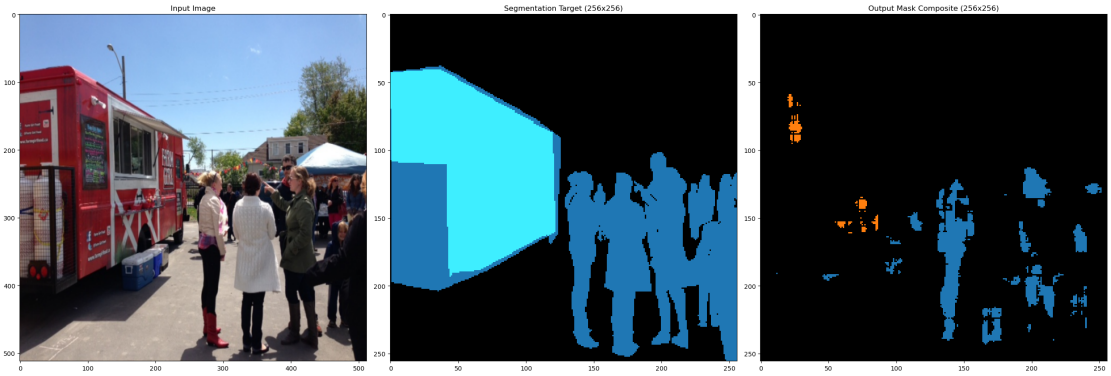

Figure 4.34: *Example of segmentation results from the InPainTor model trained for 91 classes. A food truck and several persons standing around it are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

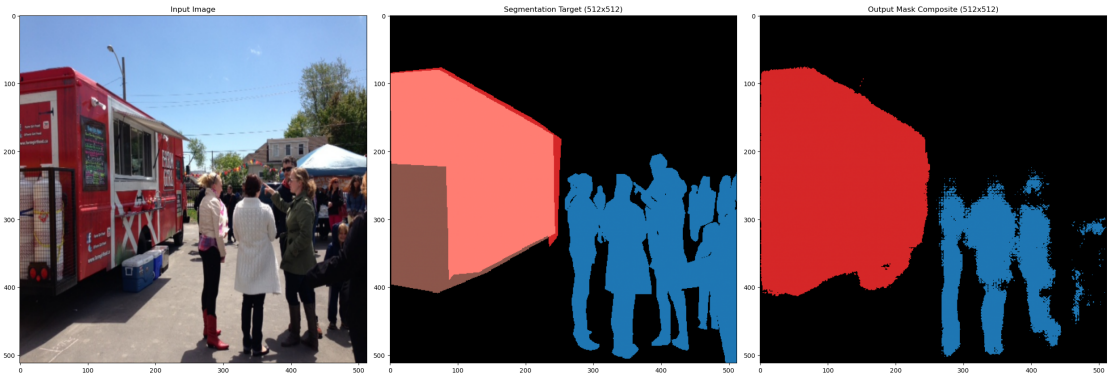

Figure 4.35: *Example of segmentation results from the ENet model trained for 16 classes. A food truck and several persons standing around it are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

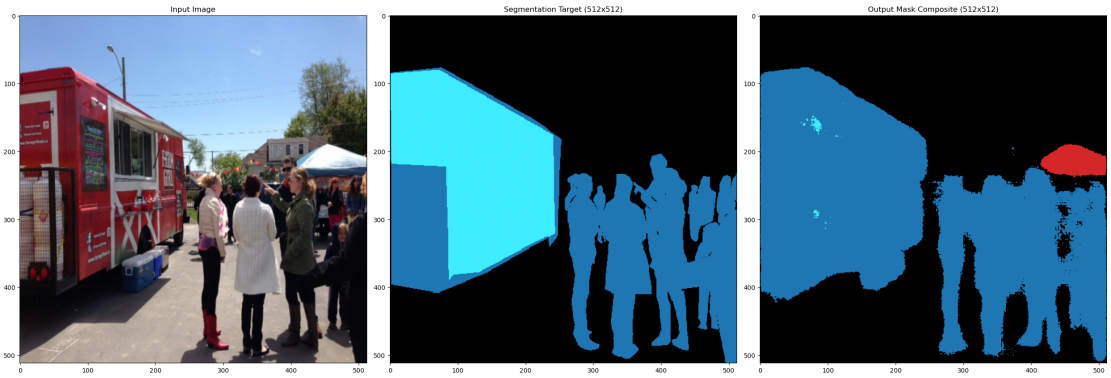

Figure 4.36: *Example of segmentation results from the InPainTor model trained for 91 classes. A food truck and several persons standing around it are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

The fourth example, depicted in Figures 4.37, 4.38, 4.39 and 4.40, shows a less represented class: airplane. The original image, in black and white, shows an airplane in midair. The InPainTor model for 16 classes cannot identify the airplane, while the InPainTor model for 91 classes only recognizes a small patch on the top of the airplane.

The tuned ENet model for 16 classes is able to segment most of the airplane. The tuned ENet model for 91 classes performs similarly, identifying a slightly larger area of the airplane.

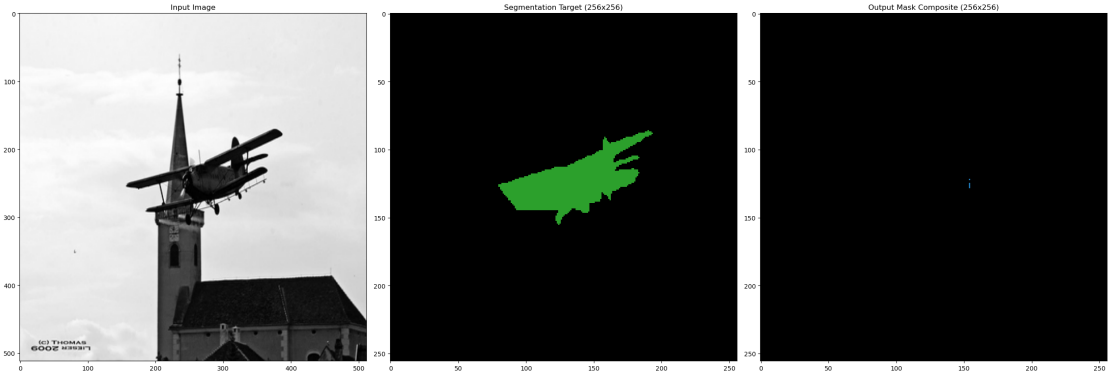

Figure 4.37: *Example of segmentation results from the InPainTor model trained for 16 classes. An airplane in midair is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

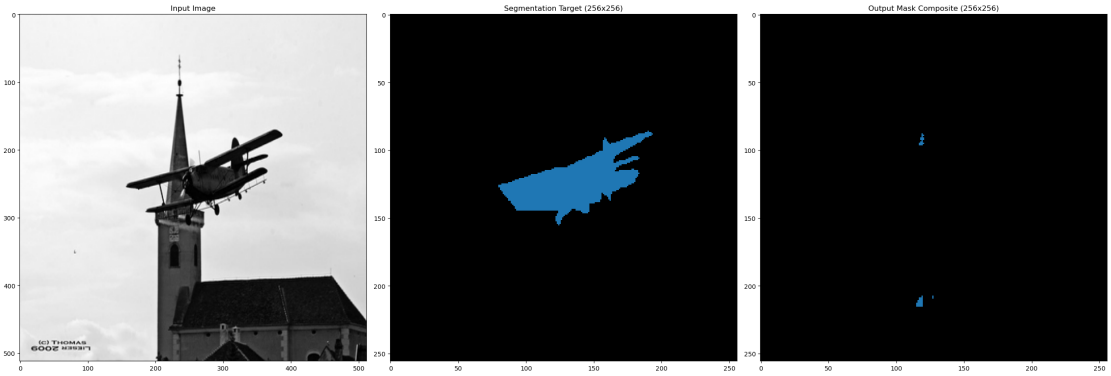

Figure 4.38: *Example of segmentation results from the InPainTor model trained for 91 classes. An airplane in midair is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

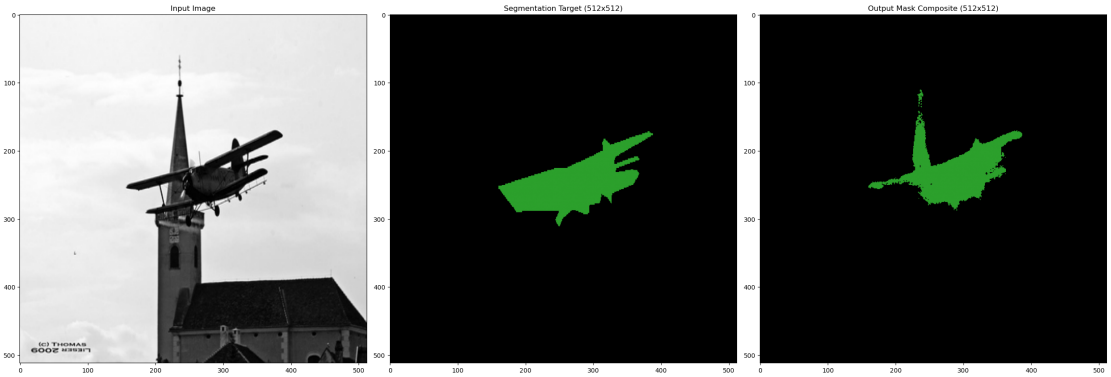

Figure 4.39: *Example of segmentation results from the ENet model trained for 16 classes. An airplane in midair is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

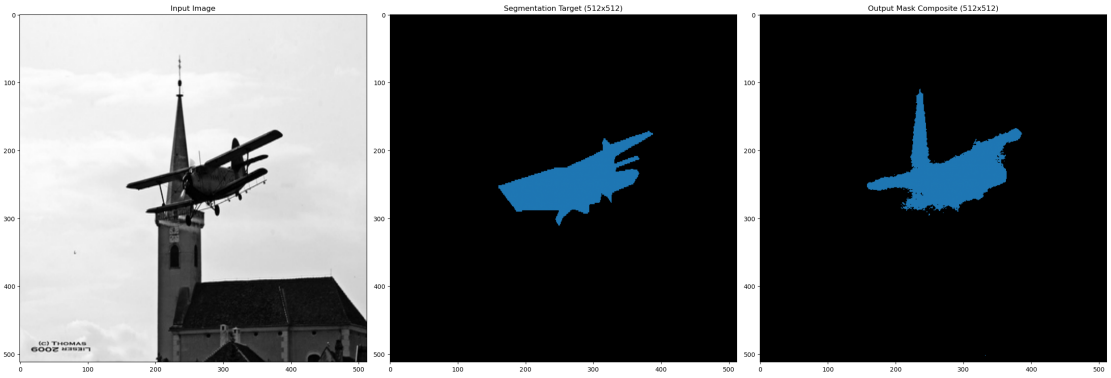

Figure 4.40: *Example of segmentation results from the InPainTor model trained for 91 classes. An airplane in midair is represented. From left to right: original image, ground truth mask, mask from segmentation model.*

The fifth example, depicted in Figures 4.41, 4.42, 4.43 and 4.44, shows a complex outdoor scenario, with multiple persons playing football and vehicles in the background. The InPainTor model for 16 classes is only able to identify some small patches of the persons, while the InPainTor model for 91 classes performs slightly better, identifying a larger area of the persons. The tuned ENet model for 16 classes performs significantly better, correctly segmenting most of the persons, as well as the vehicles in the background. The tuned ENet model for 91 classes identifies the persons somewhat better, but the vehicles are not as well segmented.

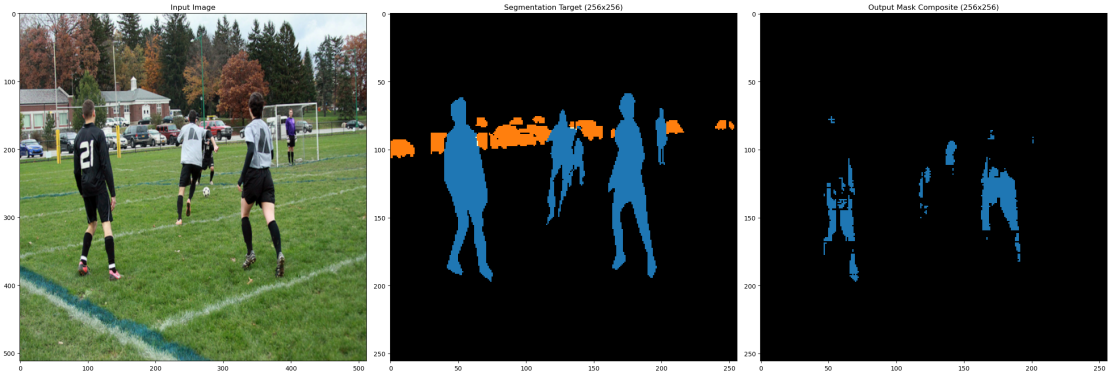

Figure 4.41: *Example of segmentation results from the InPainTor model trained for 16 classes. A complex outdoor scenario, with multiple persons playing football and vehicles in the background. From left to right: original image, ground truth mask, mask from segmentation model.*

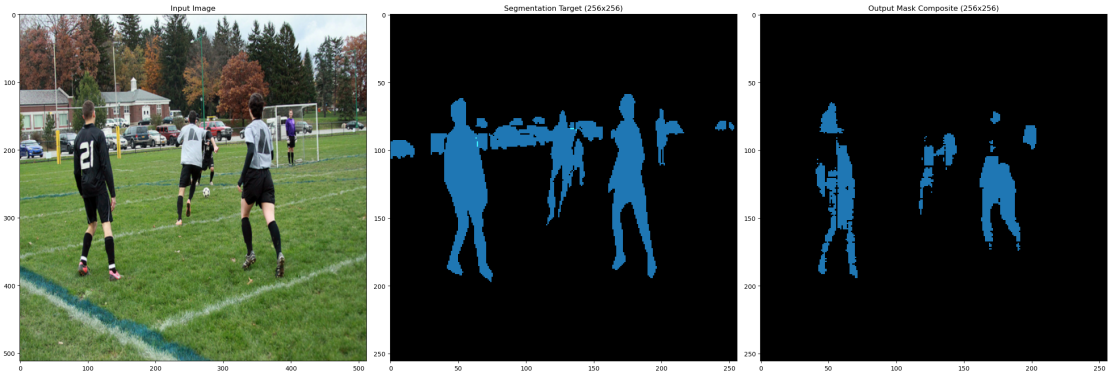

Figure 4.42: *Example of segmentation results from the InPainTor model trained for 91 classes. A complex outdoor scenario, with multiple persons playing football and vehicles in the background. From left to right: original image, ground truth mask, mask from segmentation model.*

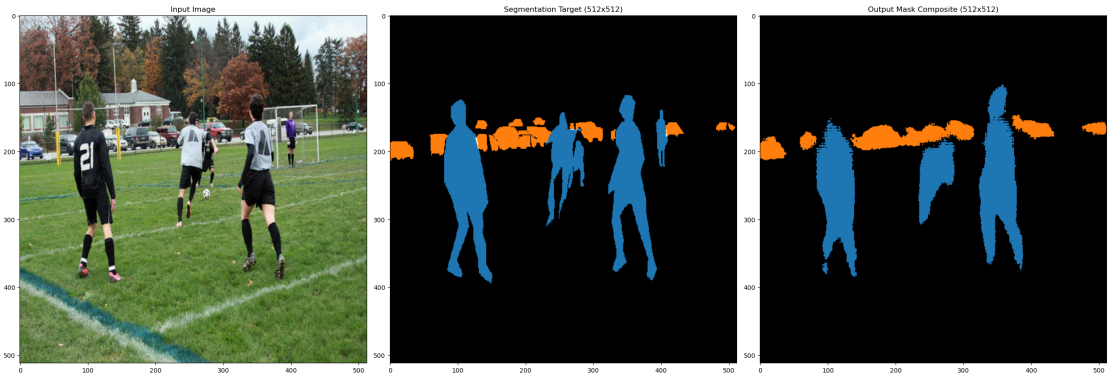

Figure 4.43: *Example of segmentation results from the ENet model trained for 16 classes. A complex outdoor scenario, with multiple persons playing football and vehicles in the background. From left to right: original image, ground truth mask, mask from segmentation model.*

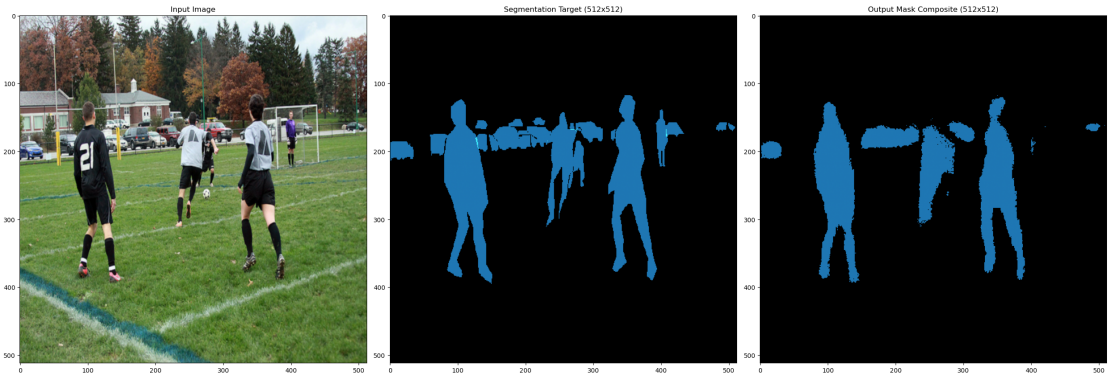

Figure 4.44: *Example of segmentation results from the InPainTor model trained for 91 classes. A complex outdoor scenario, with multiple persons playing football and vehicles in the background. From left to right: original image, ground truth mask, mask from segmentation model.*

The sixth and final example, depicted in Figures 4.45, 4.46, 4.47 and 4.48, shows two persons eating, covering a large area of the image. The InPainTor model for 16 classes is only able to identify some patches of the persons, mostly in their faces, while the InPainTor model for 91 classes identifies more patches of the body of the person the

left and less patches of the person on the right. The tuned ENet model for 16 classes segments most of the person on the left, but struggles somewhat with the person on the right. The tuned ENet model for 91 classes segments the persons better, but is unable to correctly segment the bottle or the food that the person on the left is holding.

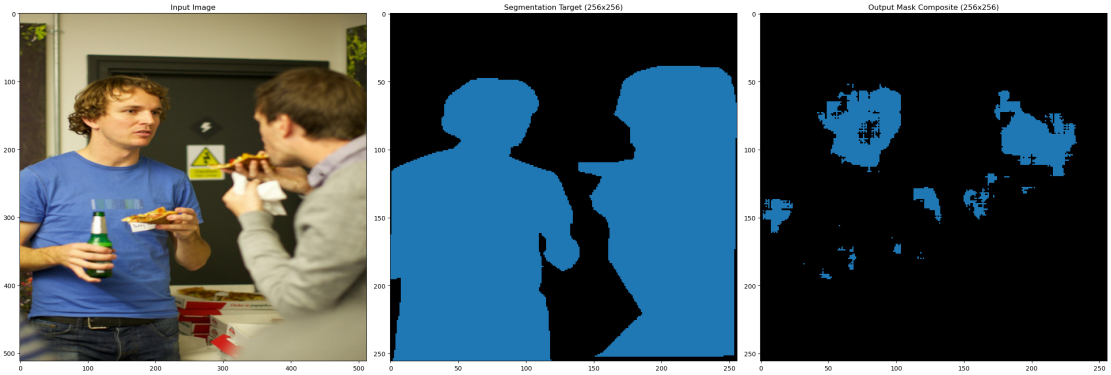

Figure 4.45: *Example of segmentation results from the InPainTor model trained for 16 classes. Two persons eating are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

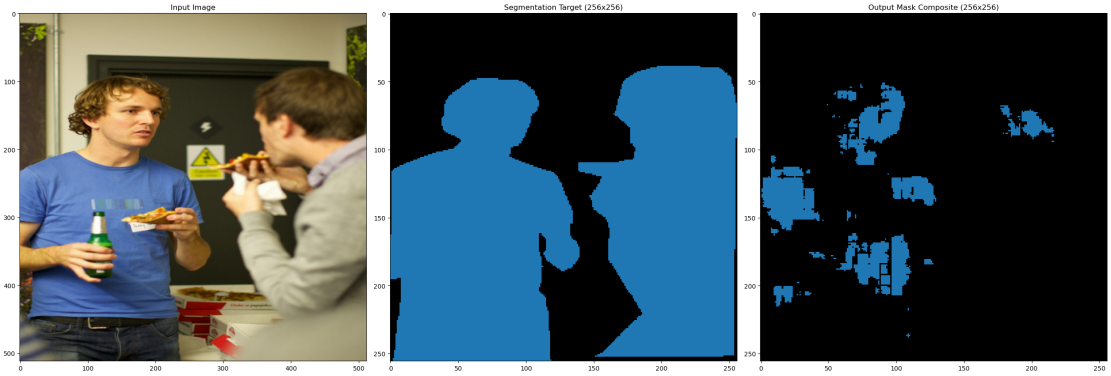

Figure 4.46: *Example of segmentation results from the InPainTor model trained for 91 classes. Two persons eating are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

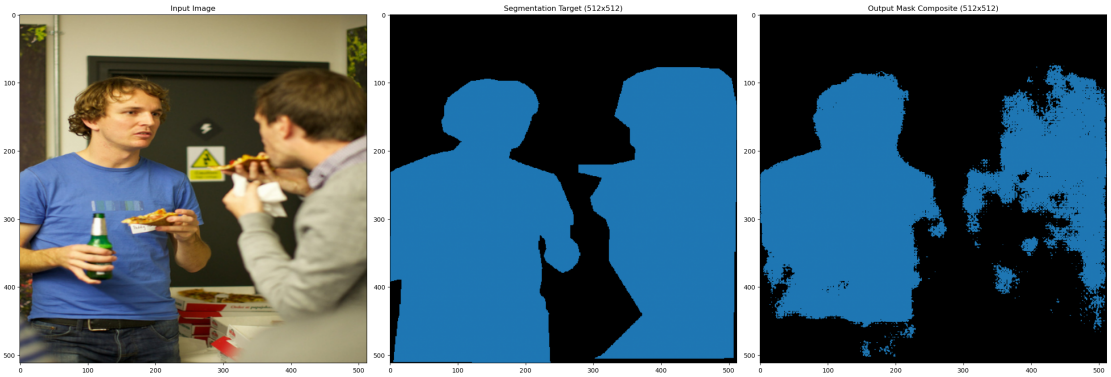

Figure 4.47: *Example of segmentation results from the ENet model trained for 16 classes. Two persons eating are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

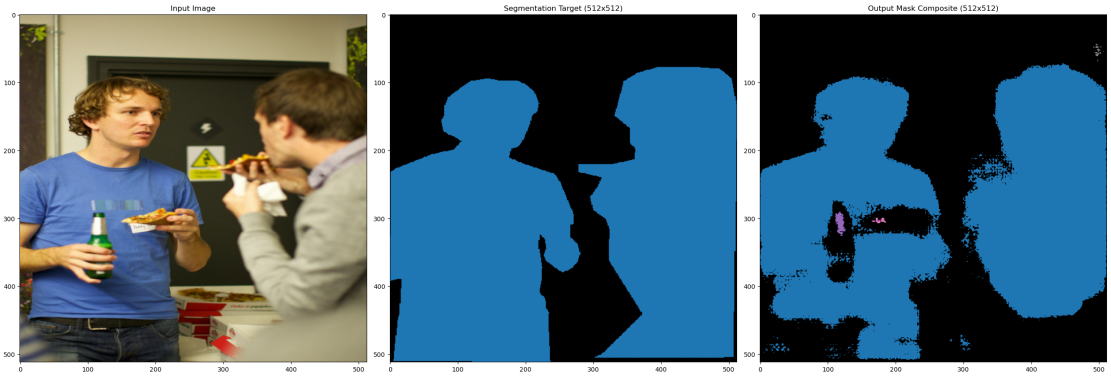

Figure 4.48: *Example of segmentation results from the InPainTor model trained for 91 classes. Two persons eating are represented. From left to right: original image, ground truth mask, mask from segmentation model.*

4.4 Inpainting Analysis

In this section, we detail the evaluation metrics and results for the inpainting models and variations of the ModInPainTor pipeline.

4.4.1 Evaluation Metrics and Results

The inpainting models were evaluated on the validation set of the RORD dataset, as well as on the RPVD. The PSNR and SSIM values were obtained for each model after normalizing the inpainted and ground truth images to a range of $[0, 255]$. The OpenCV library provided the functions to calculate these metrics. More details on these metrics are provided in Section 2.6.

The inference time of each model/pipeline was also measured, in terms of processed IPS. This metric is important to understand the efficiency of the models, especially in real-time applications. For individual models, the inference time was measured by processing the entire test set and calculating the average time the model took to process each image. For each batch, only the time taken for the model to process the input images was recorded. Similarly, for the ModInPainTor pipeline, the total time taken to process the test set was recorded, and the average time per image was computed. Here, we take into account the time taken for all processing steps, including the segmentation and inpainting stages, as well as processing tasks between stages. These processing tasks include the creation of the binary mask for the inpainting model. This approach ensures that we capture the performance of the entire system, including any overhead introduced by intermediate processing steps.

As the InPainTor model directly integrates all of its components, we cannot obtain results for its generative performance alone. However, to obtain the inference times in Table 4.4, the generative decoder was disabled, and only the shared encoder and segmentation decoder were used to process the images.

4.4.2 Results

The InPainTor, ModInPainTor and PEPSI results on the RORD validation set, including the PSNR and SSIM scores and inference times, can be seen in Table 4.5.

Table 4.5: *Pipeline Evaluation Metrics and Inference Times on the RORD Validation Set. The first row corresponds to InPainTor, the second corresponds to the PEPSI generative model given the RORD segmentation masks, while the remaining rows pertain to ModInPainTor.*

Segmentation Model	Gen. Model	PSNR	SSIM	IPS
InPainTor	InPainTor	22.7545	0.7904	**1164.2352**
RORD Segmentation Masks	PEPSI	**24.7110**	0.8580	39.5727
ENet Tuned 91 cls.	PEPSI	22.4046	0.8746	25.1276
ENet Original 91 cls.	PEPSI	20.9737	**0.8852**	27.2391

The InPainTor model obtained a PSNR score of 22.7545db and an SSIM of 0.7904 when using the RORD binary masks. It was able to process 1164.2352 IPS in our tests.

The PEPSI model obtained a PSNR score of 24.7110db and an SSIM of 0.8580 when using the RORD binary masks. As a standalone generative model, it was able to process 39.5727 IPS in our tests.

When combining the segmentation and generative models into ModInPainTor, we find that with the original ENet model, ModInPainTor yields a PSNR value of 20.9737db, an SSIM score of 0.8852 and it is able to process 27.2391 IPS. Using the tuned ENet model shows a significant improvement in object removal performance, increasing the PSNR value to 22.4046db, although the SSIM score lowers to 0.8746. The inference times are slightly increased, resulting in 25.1276 IPS.

We provide boxplots to better visualize the distribution of PSNR and SSIM values for each model/pipeline obtained on the RORD validation dataset in Figures 4.49 and 4.50. These boxplots illustrate the median, quartiles, and potential outliers in the performance metrics, offering a clearer understanding of each model's consistency and reliability across the validation dataset. Each value is calculated for each image in the validation set, and the boxplots summarize these distributions.

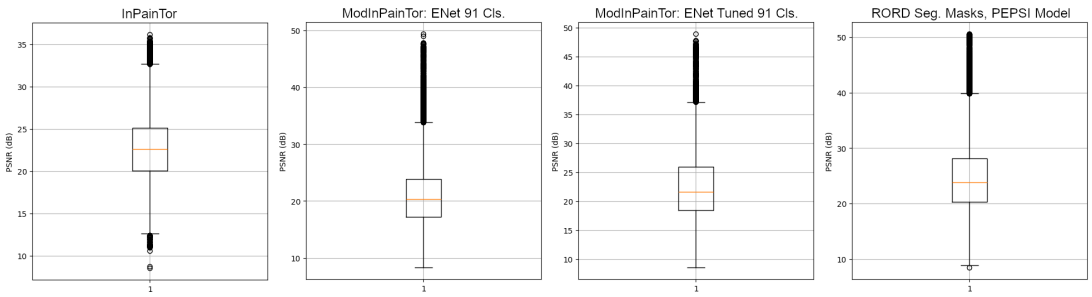

Figure 4.49: *PSNR score distributions on the RORD validation dataset.*

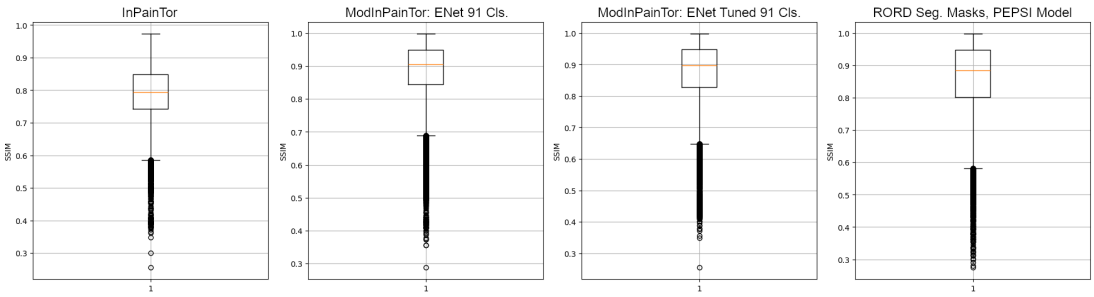

Figure 4.50: *SSIM score distributions on the RORD validation dataset.*

While the PEPSI model shows the best PSNR score, the ModInPainTor pipeline using the original ENet model trained for 91 classes shows the best SSIM score. The ModInPainTor pipeline using the tuned ENet model trained for 16 classes shows a significant improvement in the PSNR score, but a slight decrease in SSIM score when compared to the original ENet model. The InPainTor model shows a slightly higher PSNR score than the ModInPainTor pipeline using the tuned ENet model trained for 91 classes, but a significantly lower SSIM score.

The ModInPainTor results on the RPVD, including the PSNR and SSIM scores and inference times, can be seen in Table 4.6.

Table 4.6: *ModInPainTor Evaluation Metrics and Inference Times on the RPVD.*

Segmentation Model	Gen. Model	PSNR	SSIM	IPS
ENet Tuned 16 cls.	PEPSI	22.0849	0.8890	23.7919
ENet Tuned 91 cls.	PEPSI	**22.4599**	**0.8911**	24.0277
ENet Original 91 cls.	PEPSI	20.5973	0.8818	**27.1425**

On the RPVD, the ModInPainTor using the ENet tuned model trained for 16 COCO classes yields a PSNR of 22.0849db and an SSIM of 0.8890, while being able to process 23.7919 IPS. With the original ENet model trained for 91 classes, ModInPainTor yields a PSNR of 20.5973db and an SSIM of 0.8818, processing 27.1425 IPS. The tuned ENet model trained for 91 classes shows a slight increase in PSNR to 22.4599db and an increase in SSIM to 0.8911, while being able to process 24.0277 IPS.

We also provide boxplots to better visualize the distribution of PSNR and SSIM values for the variations of ModInPainTor obtained on the RPVD in Figures 4.51 and 4.52.

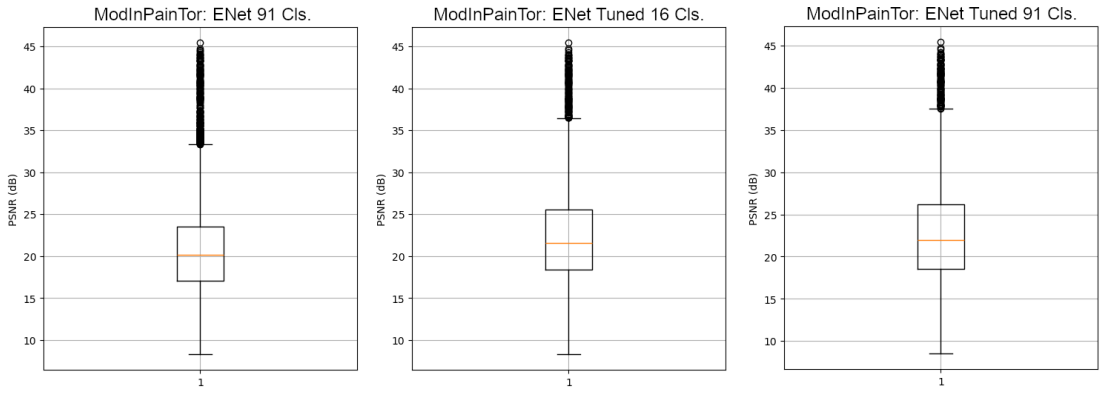

Figure 4.51: *PSNR score distributions on the RPVD.*

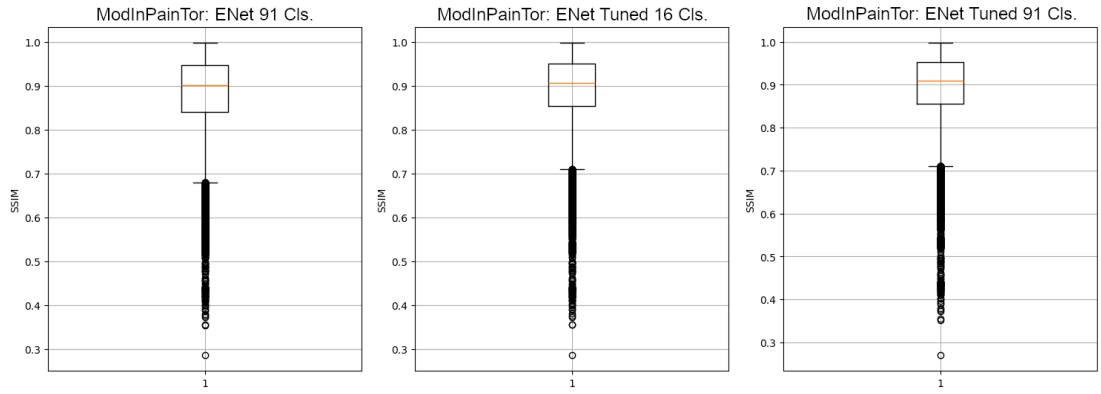

Figure 4.52: *SSIM score distributions on the RPVD.*

4.4.3 Qualitative Results

In this section, we provide qualitative results to better illustrate the performance of each model/pipeline. We divided the results into two parts: results on the RORD validation dataset, and results on the RPVD.

Results on the RORD Validation Dataset

We provide qualitative results obtained on the RORD validation dataset. We selected seven examples that illustrate the strengths and weaknesses of each model/pipeline.

The first example contains two persons playing table tennis indoors. This image particularly showcases the difficulty of accurately reconstructing non-masked regions by the InPainTor model. The ModInPainTor pipeline, using the tuned ENet models, is able to achieve better results by providing more accurate segmentation masks. However, the generative model struggles with inpainting objects where the segmentation masks do not fully cover them. Furthermore, some distortion is introduced in the tennis table as the segmentation model incorrectly identifies a patch. The generative model shows convincing results when given accurate segmentation masks.

The results obtained by the InPainTor model can be seen in Figure 4.53, while the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.54, and the results obtained using the tuned ENet model can be seen in Figure 4.55. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.56.

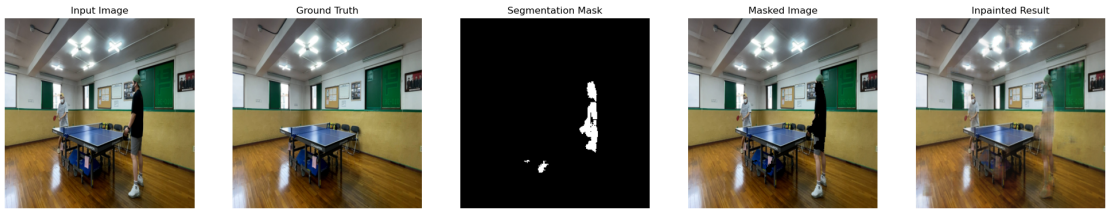

Figure 4.53: *Example of image with two persons playing table tennis indoors. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

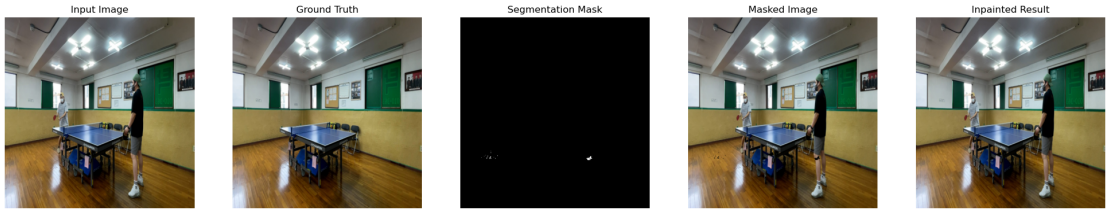

Figure 4.54: *Example of image with two persons playing table tennis indoors. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

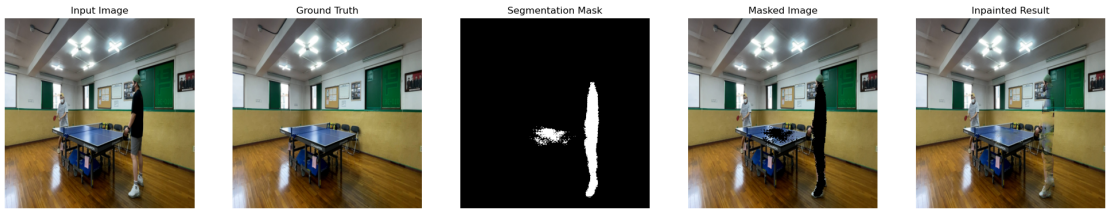

Figure 4.55: *Example of image with two persons playing table tennis indoors. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

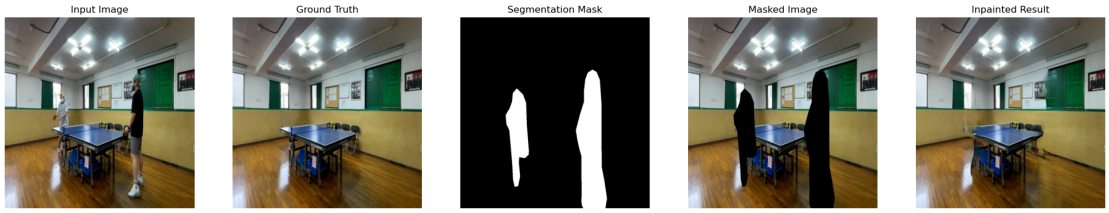

Figure 4.56: *Example of image with two persons playing table tennis indoors. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The next example contains one person standing in an art gallery. The background is more uniform than in the previous example. The ModInPainTor pipeline using the tuned ENet model produced good results, with the inpainted image closely resembling the ground truth, albeit with some artifacts where the segmentation masks failed to fully cover the person. The InPainTor appears to have blurred the person with the background colors. In terms of pixel-wise evaluation, the InPainTor model may be able to match the ModInPainTor results in some cases, but the overall quality is lower, as evidenced by the SSIM scores.

The results obtained by the InPainTor model can be seen in Figure 4.57, while the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.58, and the results obtained using the tuned ENet model can be seen in Figure 4.59. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.60.

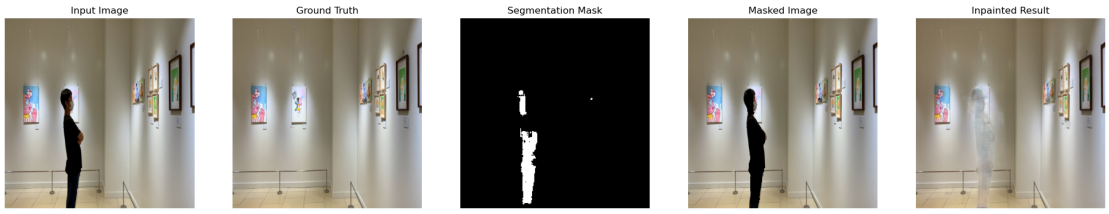

Figure 4.57: *Example of image with one person standing in an art gallery. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

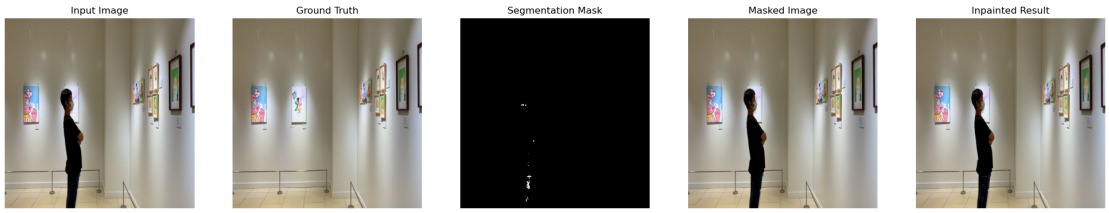

Figure 4.58: *Example of image with one person standing in an art gallery. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

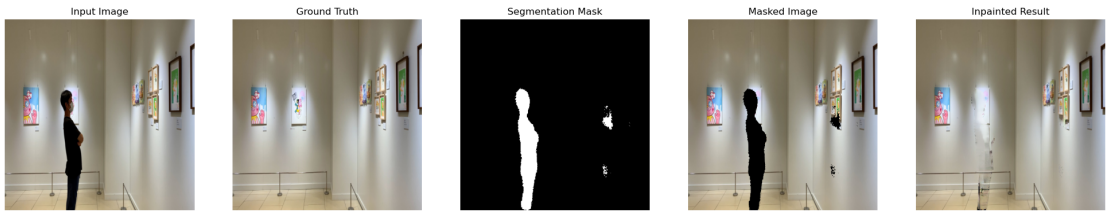

Figure 4.59: *Example of image with one person standing in an art gallery. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

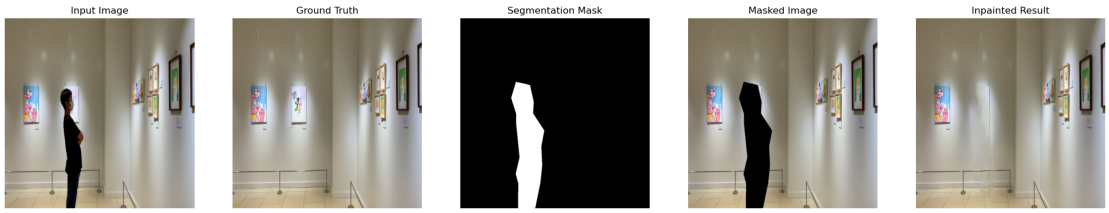

Figure 4.60: *Example of image with one person standing in an art gallery. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The third example contains two persons playing basketball outdoors. The ModInPainTor pipeline using the tuned ENet segmentation model is able to successfully remove one person from the image. In the InPainTor example, we can clearly see that the second person (with the orange shirt) has been blurred, although the segmentation mask does not cover it. This indicates that the InPainTor model may ignore the segmentation masks in some cases. Sometimes, this can be beneficial, as it attempts to remove objects that were not properly segmented. Pixel-wise, the non-identified objects become more similar to the ground truth image, resulting in a higher PSNR score. The generative model, given the original RORD binary masks, is able to produce convincing results, closely resembling the ground truth image. Some artifacts are present, but the overall quality is high.

The results obtained by the InPainTor model can be seen in Figure 4.61, while the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.62. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.64.

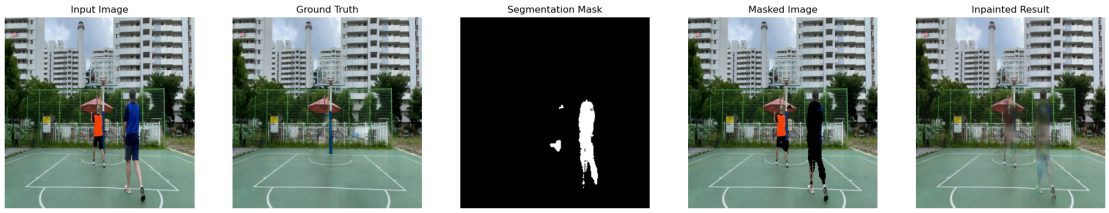

Figure 4.61: *Example of image with two persons playing basketball outdoors. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

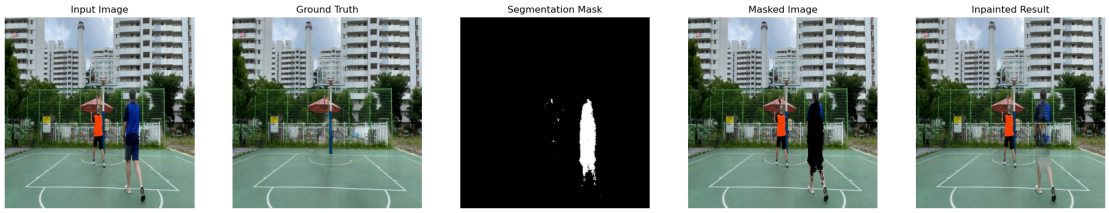

Figure 4.62: *Example of image with two persons playing basketball outdoors. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

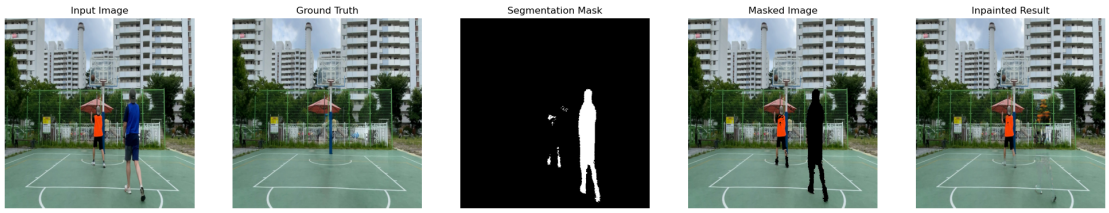

Figure 4.63: *Example of image with two persons playing basketball outdoors. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

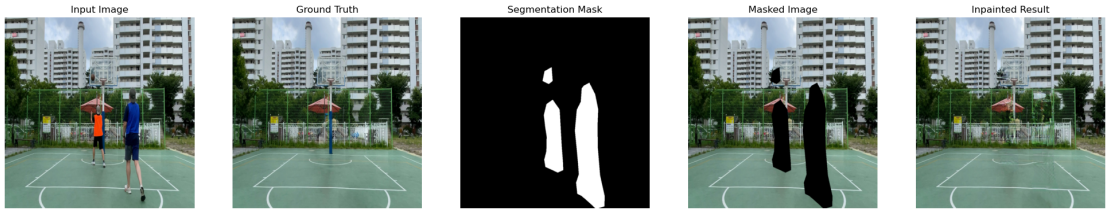

Figure 4.64: *Example of image with two persons playing basketball outdoors. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The fourth example contains two persons standing in a field of sand and grass in front of a building. One person stands near the camera, while the other is far away and therefore represents a small area of the image. The background is more complex than in the previous examples. Once again, the InPainTor model appears to blur the person in the foreground with the background colors, while the ModInPainTor pipeline using the tuned ENet model is able to produce more convincing results. A faint outline of the person can still be seen in the inpainted image, but the inpainted content resembles the ground truth image. The generative model, given the original RORD binary masks, does a stellar job at inpainting the areas covered by the people, closely resembling the ground truth image. The ENet segmentation models in the ModInPainTor pipeline were only able to segment the person in front.

The results obtained by the InPainTor model can be seen in Figure 4.65. While the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.66, and the results obtained using the tuned ENet model can be seen in Figure 4.67. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.68.

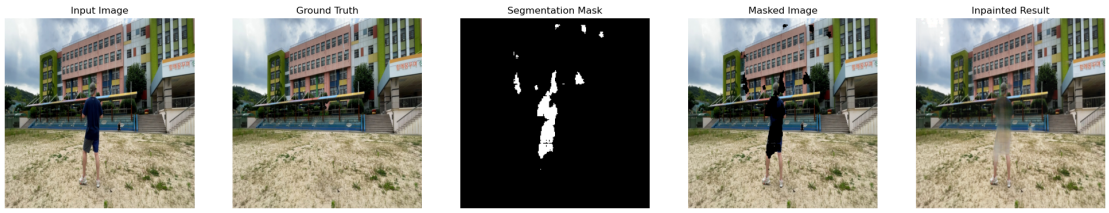

Figure 4.65: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

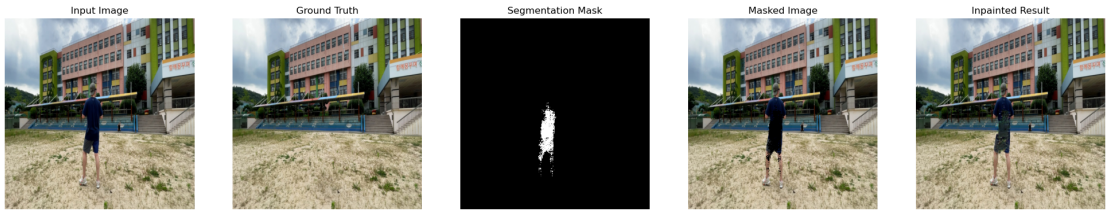

Figure 4.66: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

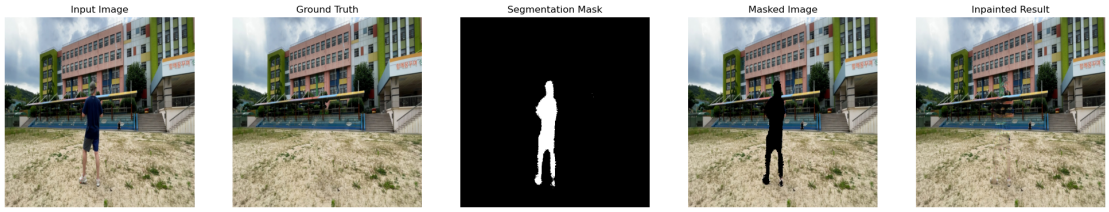

Figure 4.67: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

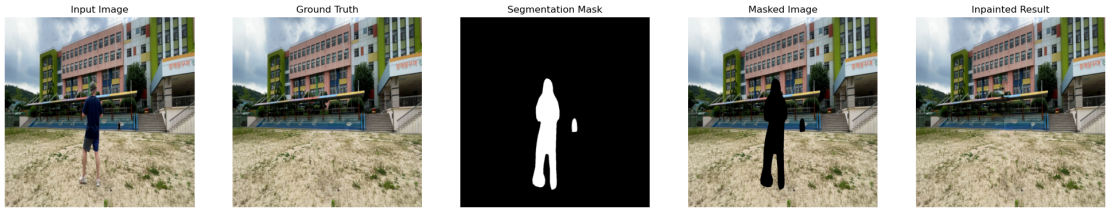

Figure 4.68: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The fourth example contains several vehicles in an underground parking lot. The vehicles should be removed from the scene. The InPainTor model struggles to both segment and inpaint the vehicles, resulting in a blurry image where the vehicles are still clearly present. The tuned ENet segmentation model is able to segment the vehicles

in the image much more effectively. However, this image is extremely challenging to inpaint, as the vehicles represent a large portion of the image.The generative model produces a blurred image where the vehicles present many artifacts.

The results obtained by the InPainTor model can be seen in Figure 4.69. The results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.70. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.72.

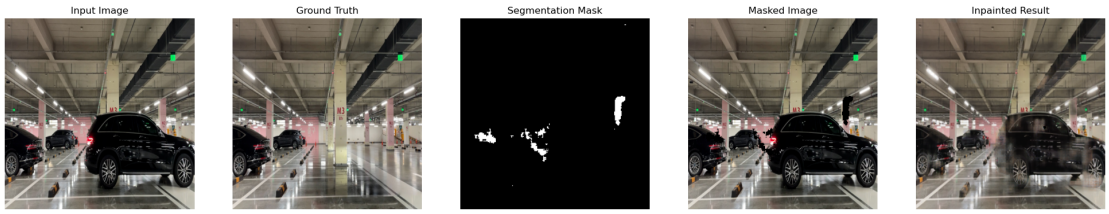

Figure 4.69: *Example of image with several vehicles in an underground parking lot. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

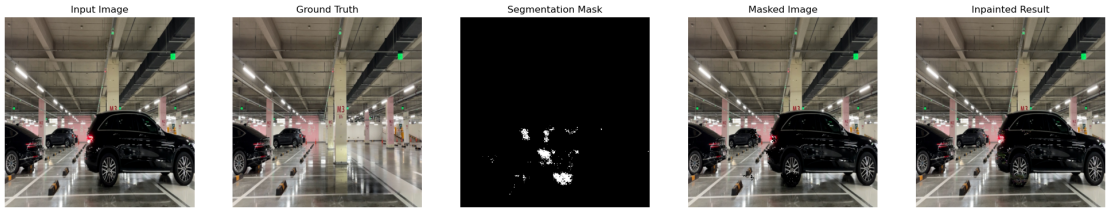

Figure 4.70: *Example of image with several vehicles in an underground parking lot. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

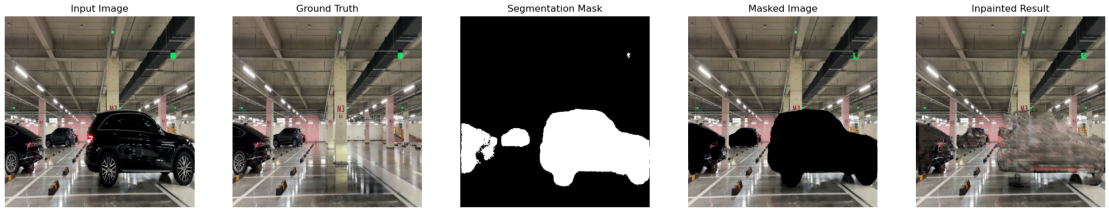

Figure 4.71: *Example of image with several vehicles in an underground parking lot. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

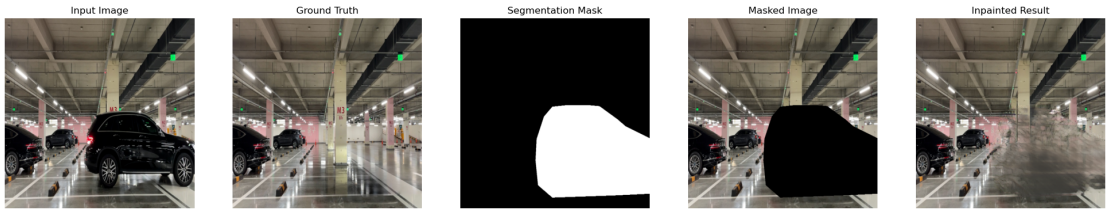

Figure 4.72: *Example of image with several vehicles in an underground parking lot. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The fifth example contains a pool table indoors. Here, the balls should be removed from the table. This image reveals the limitations of the segmentation models on smaller, less prominent objects. The InPainTor model segments several patches of the image, although the balls are not identified. The InPainTor model goes beyond the segmentation mask and appears to blur the balls with the table colors, but many artifacts are introduced in the process. The ModInPainTor pipeline was not able to segment the balls. The pipeline with the tuned ENet model segmented several patches of the image, similarly to the InPainTor, but the balls were not segmented. As such, the inpainted result maintains the balls, but adds some blurriness to other areas of the image which should not have been altered. This may negatively affect the SSIM score, as the structural similarity is compromised. Pixel-wise, the inpainted areas closely match the ground truth image and should therefore not significantly affect the PSNR score. The generative model, given the original RORD binary masks, is able to produce convincing results, closely resembling the ground truth image.

The results obtained by the InPainTor model can be seen in Figure 4.73, while the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.74. We also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.76.

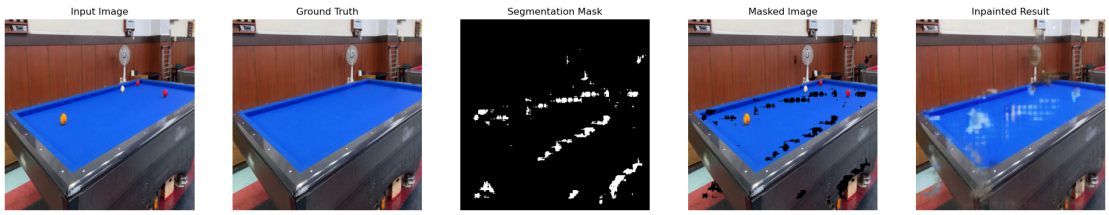

Figure 4.73: *Example of image with pool table. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

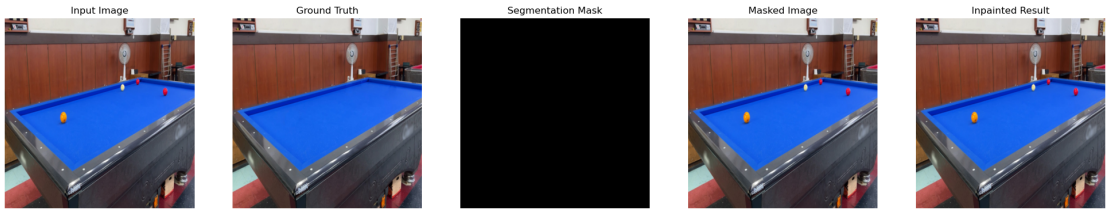

Figure 4.74: *Example of image with pool table. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

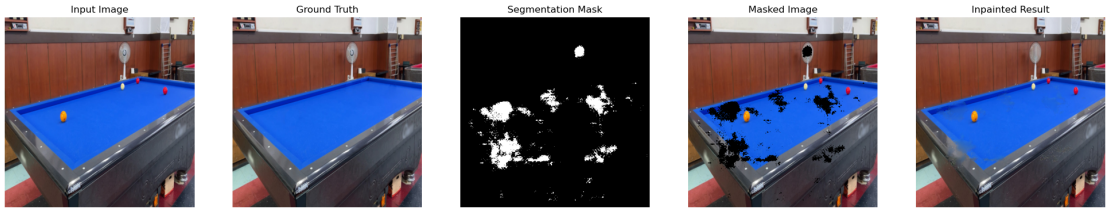

Figure 4.75: *Example of image with pool table. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

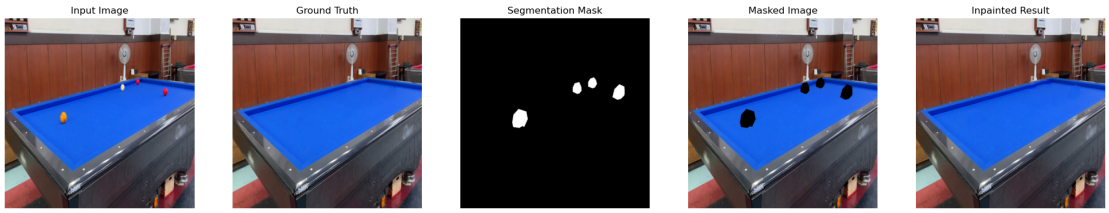

Figure 4.76: *Example of image with pool table. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

The last example contains one person in a mall. The person is far away from the camera. This image reveals the limitations of the models when dealing with small objects, particularly when it comes to segmentation. In this case, the InPainTor model was able to partially segment the person, resulting in a blurred inpainted image where the person is still faintly visible. The ModInPainTor pipeline, using the tuned ENet model, was not able to segment the person at all, and actually identified a large patch of the escalator, resulting in an inpainted image where the person is still completely unaffected, but the escalator is blurred. This negatively impacts the overall quality of the inpainting, and the PSNR and SSIM scores. The generative model, given the original RORD binary masks, is able to completely remove the person, but some blurriness is introduced in the inpainted area.

The results obtained by the InPainTor model can be seen in Figure 4.77, while the results obtained by the ModInPainTor pipeline, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting, can be seen in Figure 4.78, and the results obtained using the tuned ENet model can be seen in Figure 4.79. We

also obtained results directly from the PEPSI model, using the ground truth binary masks from the RORD dataset. These results can be seen in Figure 4.80.

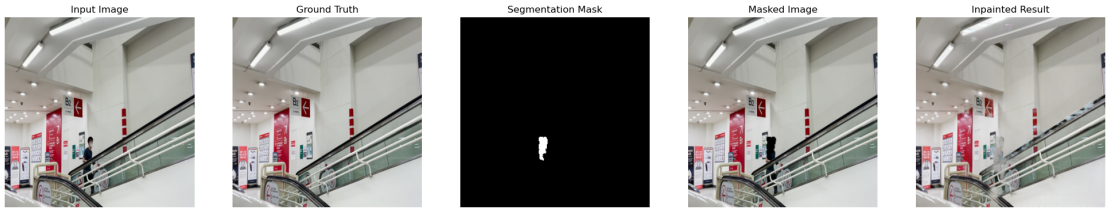

Figure 4.77: *Example of image with one person in a mall. The person is far away. Obtained from InPainTor on the RORD validation dataset. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

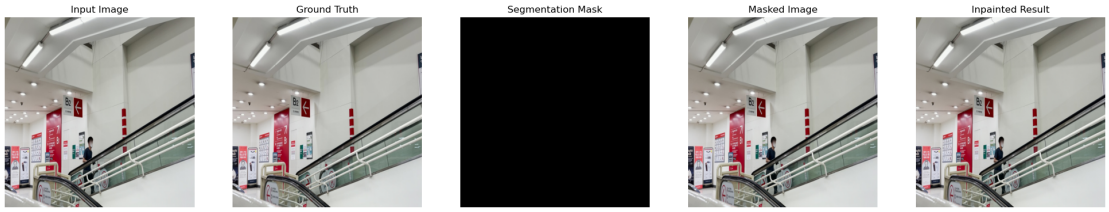

Figure 4.78: *Example of image with one person in a mall. The person is far away. Obtained from ModInPainTor, using the original ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

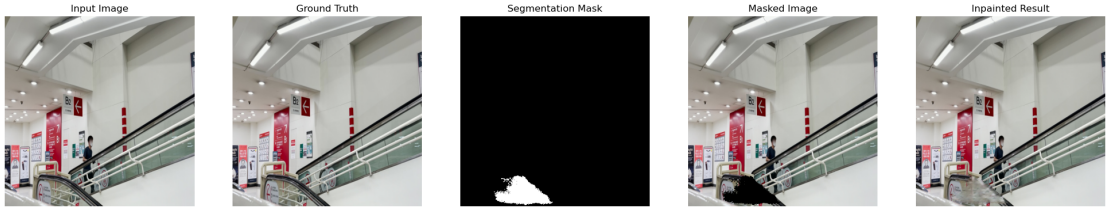

Figure 4.79: *Example of image with one person in a mall. The person is far away. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

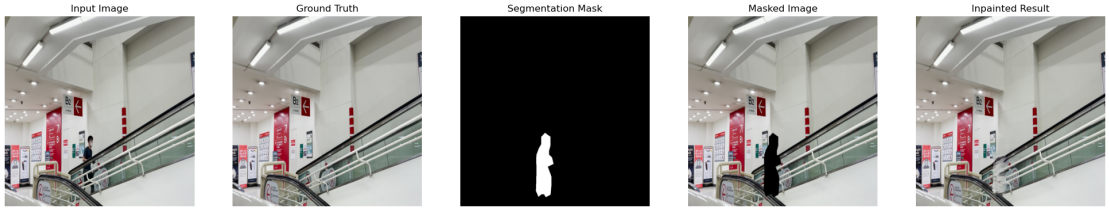

Figure 4.80: *Example of image with one person in a mall. The person is far away. Obtained from the PEPSI generative model, using RORD's original binary masks. From left to right: original image, ground truth, segmentation mask, masked image, inpainted image.*

Results on the RPVD

We provide qualitative results obtained on the RPVD. We selected some examples that illustrate the differences between the ModInPainTor pipeline variations regarding the

segmentation model. These examples were also used in 4.4.3. Here, we compare the results obtained using the fine-tuned ENet models trained for 16 and 91 classes. The PEPSI generative model is used for inpainting in all cases.

The first example shows two persons playing table tennis indoors. The ModInPainTor pipeline, using the tuned ENet model for 91 classes, is able to achieve better results thanks to the more accurate segmentation masks. While the tuned ENet model for 91 classes segments most of the person on the front, the model for 16 classes only identifies the person's upper body. However, both models failed to segment the person in the back. The generative model shows convincing results in inpainting the person in the foreground given the accurate segmentation mask from the 91 classes segmentation model. The results can be seen in Figures 4.81 and 4.82.

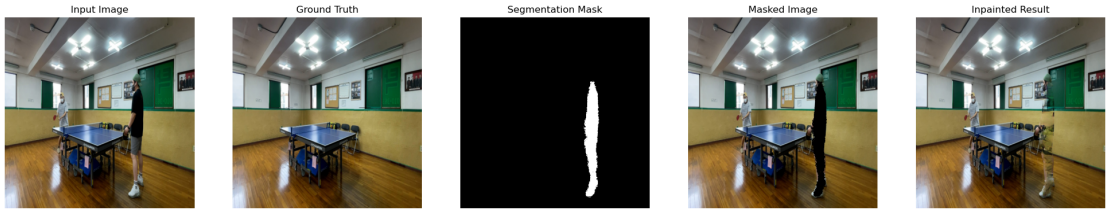

Figure 4.81: *Example of image with two persons playing table tennis indoors. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

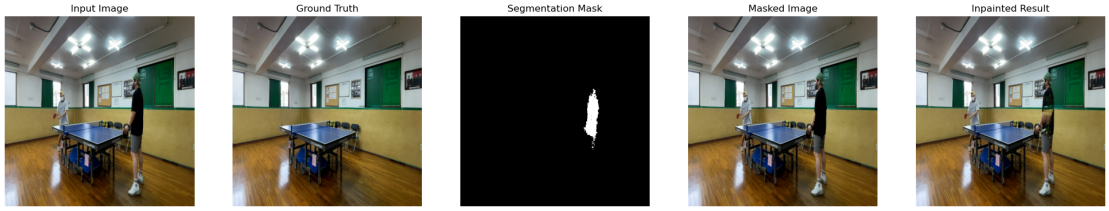

Figure 4.82: *Example of image with two persons playing table tennis indoors. Obtained from ModInPainTor, using the tuned ENet model trained for 16 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

The second example shows two persons playing basketball outdoors. Both segmentation models were able to segment the person in the foreground, but neither model correctly segmented the person in the back. Furthermore, the tuned ENet model for 91 classes produced a more accurate segmentation mask for the person in the foreground compared to the model for 16 classes. The generative model was able to produce convincing results inpainting the area covered by the person in the foreground, given the accurate segmentation mask from the 91 classes segmentation model. The results can be seen in Figures 4.83 and 4.84.

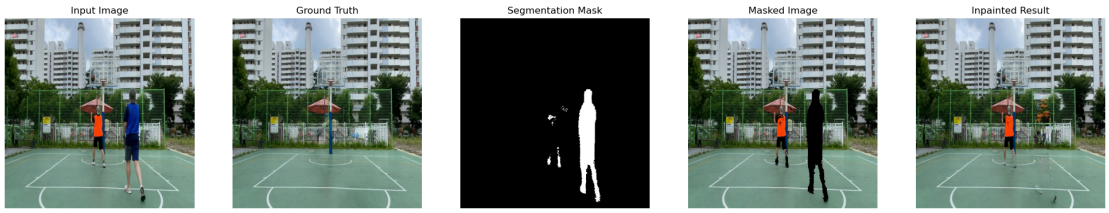

Figure 4.83: *Example of image with two persons playing basketball outdoors. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

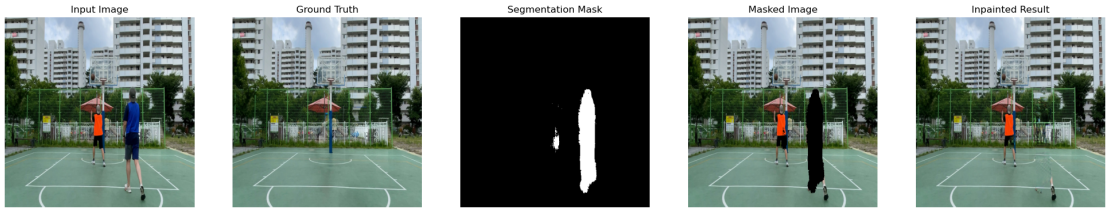

Figure 4.84: *Example of image with two persons playing basketball outdoors. Obtained from ModInPainTor, using the tuned ENet model trained for 16 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

The third example shows two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. The tuned ENet model for 91 classes was able to segment the person in the foreground slightly better than the model for 16 classes, but both models could have performed better. Neither model was able to segment the person in the back. The generative model was unable to fully inpaint the area covered by the person in the foreground, given the segmentation masks from both models. The results can be seen in Figures 4.85 and 4.86.

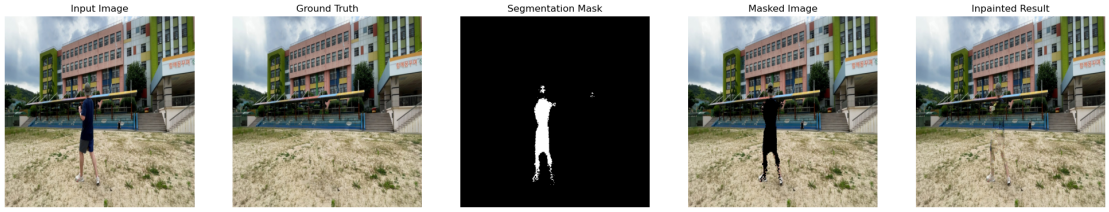

Figure 4.85: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

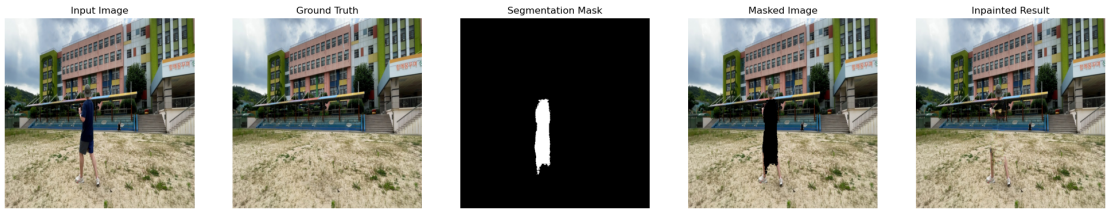

Figure 4.86: *Example of image with two persons standing in a field in front of a building. One is standing near the camera, and the other is far away. Obtained from ModInPainTor, using the tuned ENet model trained for 16 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

The fourth example shows one person loading a small pickup truck. The tuned ENet model for 91 classes was only able to segment part of the person, while the model for 16 classes was unable to segment anything. The generative model was unable to inpaint the area covered by the person, given the segmentation masks from both models. The result on the 91 classes segmentation model shows some blurring in the area where the person was located. The results can be seen in Figures 4.87 and 4.88.

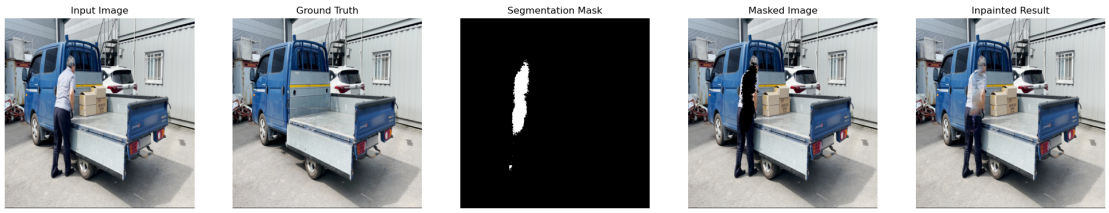

Figure 4.87: *Example of image with one person loading a small pickup truck. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

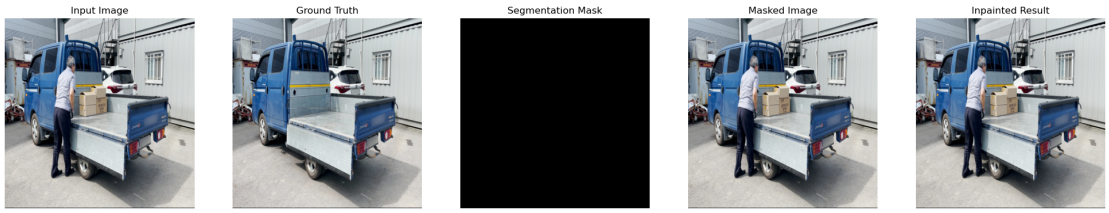

Figure 4.88: *Example of image with one person loading a small pickup truck. Obtained from ModInPainTor, using the tuned ENet model trained for 16 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

In the fifth and final example, we have one person walking near some benches under a wooden structure. The background is very challenging, as the person blocks a wooden post, some benches, and greenery. The tuned ENet model for 91 classes was able to segment a large area of the person, but missed a leg and a foot. The model for 16 classes segmented a significantly smaller area of the person. The generative model struggled to inpaint the area covered by the person, given the segmentation masks from both models. Both results show significant blurring in the segmented areas. The results can be seen in Figures 4.89 and 4.90.

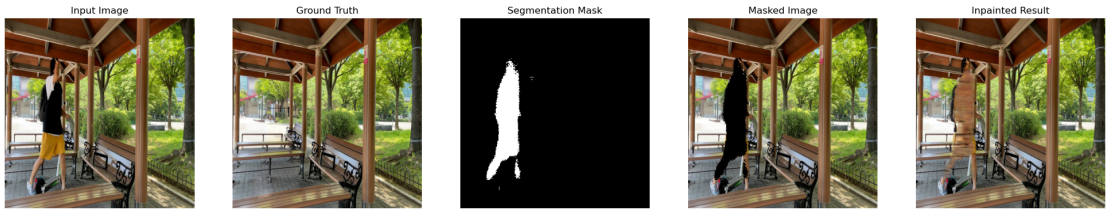

Figure 4.89: *Example of image with one person walking near some benches under a wooden structure. Obtained from ModInPainTor, using the tuned ENet model trained for 91 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

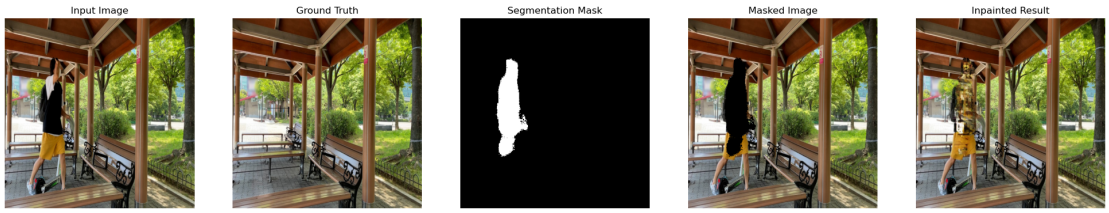

Figure 4.90: *Example of image with one person walking near some benches under a wooden structure. Obtained from ModInPainTor, using the tuned ENet model trained for 16 classes for segmentation, and PEPSI for inpainting. From left to right: original image, segmentation mask, masked image, inpainted image.*

4.5 Discussion

The results presented in this chapter demonstrate the effectiveness of the ModInPainTor pipeline in improving object removal performance compared to the InPainTor model. The quantitative metrics, specifically SSIM, indicate that the ModInPainTor pipeline achieves superior structural similarity to the ground truth images (ModInPainTor obtained SSIM scores above 0.87, while InPainTor scored below 0.80). This suggests that the ModInPainTor pipeline is more effective at preserving the overall structure and context of the images during the inpainting process.

The segmentation models based on ENet (Paszke et al., 2016) showed a marked improvement in performance over the InPainTor segmentation decoder. The tuned models achieved a substantially improved mIoU when compared to both the original ENet model (a difference of 12.922%, or nearly 2.25× for the 91 classes models) and the InPainTor segmentation decoder (a difference of 15.602%, or over 5× improvement for the 91 class models), resulting qualitatively in much more consistent and accurate segmentation masks. This improvement in segmentation accuracy is crucial for the subsequent inpainting process, as it directly affects the quality of the masks used for object removal. The segmentation results for the tuned ENet models, particularly the one trained for 16 classes, were notably better at accurately identifying and segmenting objects in the images, leading to more precise masks for the inpainting stage.

The inpainting results further highlight the benefits of using the ModInPainTor pipeline. The PEPSI model (M.-c. Sagong et al., 2019), when provided with accurate segmentation masks from the tuned ENet models, produced higher-quality inpainted

images that more closely resemble the ground truth, when compared to the InPainTor model. The quantitative metrics show that the ModInPainTor pipeline significantly improves the SSIM score (from 0.79 to 0.87), indicating better structural similarity to the original images. However, the PSNR values were slightly lower than those obtained by the InPainTor model (22.4db vs. 22.75db), which may be attributed to the nature of the segmentation masks and the inpainting process. The qualitative results support these findings, with the ModInPainTor pipeline producing visually appealing inpainted images with fewer artifacts and better integration into the surrounding context. While the InPainTor model struggles with artifacts and unrealistic inpainted areas, the ModInPainTor pipeline, particularly when using the tuned ENet model for 91 classes, demonstrates a more coherent and natural appearance in the inpainted regions. This confirms the improvement of the SSIM scores when compared to the InPainTor model. The slight decrease in PSNR can be explained by the presence of minor artifacts in some inpainted images, which may not significantly affect the overall structural similarity but can impact pixel-wise accuracy. The InPainTor model often produced blurred inpainted areas, which, while reducing pixel-wise errors, did not contribute to a realistic appearance.

Furthermore, the InPainTor model exhibited difficulties in reconstructing the non-masked regions of the image, leading to artifacts and distortions in the areas where no objects were removed. In contrast, the ModInPainTor pipeline maintained the integrity of the non-masked regions, resulting in a more consistent and visually pleasing output. This may be attributed to the fact that the InPainTor's generative decoder was trained on the masks generated by its own segmentation decoder, which were of lower quality than those produced by the tuned ENet models. This discrepancy likely led to the generative decoder struggling to accurately reconstruct the non-masked areas, as it was not exposed to high-quality masks during training. In some cases, we can clearly identify that the generative decoder of InPainTor attempted to inpaint areas that were not masked. In some cases, this can be beneficial, as it can compensate for inaccuracies in the segmentation mask. However, in most cases, this leads to artifacts and distortions in the non-masked areas of the image, which detracts from the overall quality of the inpainted image. As the PEPSI model was trained directly on the binary masks provided by the RORD dataset, it is able to effectively inpaint the masked areas without affecting the non-masked regions, leading to a more coherent and visually appealing result.

The inference times of the ModInPainTor pipeline were significantly higher than those of the InPainTor model (InPainTor was able to process 1164 IPS, while ModInPainTor could not process more than 27 IPS), primarily due to the additional complexity of each of the models. In the segmentation analysis, we can observe that the tuned ENet models require more computational resources than the original ENet model (45.02 GFLOPS vs. 5.5 GFLOPS), but processing times remained similar (14.1 IPS vs. 13.5 IPS). The PEPSI generative model is much heavier computationally than the InPainTor model (estimated at 85 GFLOPS with the tuned ENet segmentation model, vs. 20

GFLOPS). The ModInPainTor pipeline, while slower than the InPainTor model, still achieved reasonable processing speeds, making it suitable for practical applications where real-time performance is not a strict requirement (25 IPS vs. 1164 IPS). The trade-off between speed and quality is evident, with the ModInPainTor pipeline prioritizing higher-quality results at the expense of increased inference times. Furthermore, as hardware continues to advance, the computational demands of the ModInPainTor pipeline may become less of a concern, making it a more viable option for a wider range of applications.

4.6 Future Directions

The ModInPainTor pipeline demonstrates a significant advancement in object removal capabilities, particularly in terms of structural similarity and visual quality. The integration of high-quality segmentation models and a robust generative inpainting model has proven effective in addressing the limitations of the InPainTor model. However, there are several areas for future research and improvement:

- **Segmentation Model:** Improving the segmentation model's accuracy is crucial for achieving better inpainting results, as per our findings. The generative model relies heavily on the quality of the segmentation masks. Exploring more advanced segmentation architectures, such as DeepLabV3+ (L.-C. Chen et al., 2018), could result in more precise object boundaries and improved mask quality, thus significantly improving inpainting quality.
- **Generative Model:** The PEPSI model (M.-c. Sagong et al., 2019) we used for our experiments shows promising results, but there are situations where it struggles to produce realistic results, especially when large areas need to be inpainted. This is a common issue with GAN-based models. While diffusion-based models are generally more computationally intensive, and more prone to hallucination, they have shown superior performance in generating high-quality images in complex scenarios. Exploring diffusion-based inpainting models could lead to significant improvements in the realism and coherence of the inpainted areas. The ModInPainTor architecture, given its modularity, would allow for the easy integration of such models.
- **Data Augmentation:** Implementing advanced data augmentation techniques during training could enhance the robustness and generalization capabilities of both the segmentation and inpainting models. Techniques such as random cropping, rotation, color jittering and noise injection could help the models better handle diverse scenarios and improve their performance on unseen data.
- **Evaluation Metrics:** Exploring additional evaluation metrics that capture perceptual quality and user satisfaction could provide a more comprehensive assessment of inpainting performance. In our tests, there were instances where the PSNR and SSIM metrics evolved inversely.Metrics such as LPIPS or user studies

could offer a better perspective into the effectiveness of the models.

5 Conclusions

This project report presented a comprehensive study on the application of DL techniques for object removal in images.

Throughout the research, we followed the DSR methodology (Hevner et al., 2004; Peffers et al., 2007), which guided us through the process of identifying the problem, designing and developing a solution, demonstrating its utility, and evaluating its performance.

We evaluated the InPainTor model, which served as a benchmark for our proposed solution, ModInPainTor. ModInPainTor is a modular pipeline that allows the combination of different segmentation and inpainting models to achieve high-quality object removal in images.

For ModInPainTor, we selected the ENet model (Paszke et al., 2016) for segmentation and the PEPSI model (M.-c. Sagong et al., 2019) for inpainting. ENet is a segmentation architecture for real-time image segmentation, while PEPSI is a GAN-based generative model designed for high-quality image inpainting. Both models were trained and fine-tuned on relevant datasets (COCO (Caesar et al., 2018) for segmentation, RORD (M.-C. Sagong et al., 2022) for inpainting) to optimize their performance for the object removal task. The ENet model was fine-tuned using Optuna optimize its hyperparameters. The PEPSI model was trained using the same hyperparameters as the original authors, as they had already been optimized for the task.

Thanks to fine-tuning the ENet model, we achieved a significant improvement (5×) in segmentation mIoU compared to the original model used in InPainTor. The PEPSI model also demonstrated strong performance in inpainting tasks, producing high-quality results. The segmentation models were evaluated qualitatively using the mIoU, Accuracy, Precision, Recall and F1 Score metrics, while the inpainting model was evaluated using PSNR and SSIM.

We developed a new dataset, RPVD, based on RORD, to evaluate the anonymization capabilities of the ModInPainTor pipeline. The dataset was created by synthesizing new ground truth images for the RORD dataset, where only the person class was removed. This dataset allowed us to assess the performance of the pipeline in removing

people from images, which is a common requirement for anonymization tasks.

We compared the performance of segmentation models trained for different numbers of classes. We found that training a segmentation model for fewer classes (16 classes in COCO) led to improved segmentation metrics, but did not significantly enhance the overall inpainting quality. In the ModInPainTor pipeline, the ENet model trained for all 91 classes in COCO produced slightly better inpainting results on the RPVD when compared to the ENet model trained for 16 classes, as measured by PSNR (22.0849db vs. 22.4599db)and SSIM (0.8890 vs. 0.8911), compared to the model trained for 16 classes.

On the RORD dataset, the ModInPainTor pipeline achieved a PSNR of 22.4db and a SSIM of 0.87, while the InPainTor model achieved a PSNR of 22.75db and a SSIM of 0.79. These results indicate that ModInPainTor produces images with better structural similarity to the original images, while InPainTor achieves slightly higher pixel-wise accuracy. Qualitatively, ModInPainTor produced more visually appealing results, with fewer artifacts and better integration into the surrounding context. Improving the segmentation accuracy would likely lead to even better inpainting results, as the quality of the segmentation masks directly affects the performance of the inpainting model. To prove this claim, we tested ModInPainTor with ground truth segmentation masks, achieving a PSNR of 24.7db and a SSIM of 0.86. While the SSIM value was slightly lower than that achieved with the ENet segmentation model, the PSNR value was significantly higher. Our qualitative analysis also confirmed that using ground truth masks led to more realistic and coherent inpainted images.

Additionally, we measured the inference time of each model and the entire pipeline. The new models were able to process images at a reasonable speed (over 25 IPS), making them suitable for practical applications. Despite being significantly slower than InPainTor (over 1100 IPS), the new pipeline still achieved a processing rate which is acceptable for many real-time applications, such as anonymization tasks in industrial settings. As hardware continues to improve, the additional computational requirements of ModInPainTor may become less of a concern.

In conclusion, this project demonstrated that by using a more accurate segmentation model and a more advanced inpainting architecture, it is possible to significantly improve the performance of object removal in images. Our results show a significant leap in the quality of the inpainted images when compared to the original InPainTor model, if provided with accurate segmentation masks. Despite the improvements on the segmentation model, its limitations still affect the overall performance. As our ModInPainTor pipeline is modular, it will allow for easy integration of more robust segmentation and inpainting models in the future. As it stands, ModInPainTor is a viable solution for object removal in images, particularly in scenarios where anonymization quality is prioritized over computational efficiency.

We propose that future work explores further improvements in segmentation accuracy, either by using more advanced models or by incorporating additional training data. Additionally, exploring other inpainting architectures, such as ones based on

diffusion techniques, may lead to even better results. Finally, optimizing the inference time of the models could make them even more suitable for real-time applications, especially when using hardware with less computational power.

This page intentionally left blank.

Bibliography

Ahlawat, Harsh, Naveen Aggarwal, and Deepti Gupta (2025). "Automatic Speech Recognition: A survey of deep learning techniques and approaches". In: *International Journal of Cognitive Computing in Engineering* 6, pp. 201–237. ISSN: 2666-3074. DOI: https://doi.org/10.1016/j.ijcce.2024.12.007. URL: https://www.sciencedirect.com/science/article/pii/S2666307424000573.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). *Wasserstein GAN*. arXiv: 1701.07875 [stat.ML]. URL: https://arxiv.org/abs/1701.07875.

Bolya, Daniel et al. (2019). *YOLACT: Real-time Instance Segmentation*. arXiv: 1904.02689 [cs.CV]. URL: https://arxiv.org/abs/1904.02689.

Caesar, Holger, Jasper Uijlings, and Vittorio Ferrari (2018). *COCO-Stuff: Thing and Stuff Classes in Context*. arXiv: 1612.03716 [cs.CV]. URL: https://arxiv.org/abs/1612.03716.

Carion, Nicolas et al. (2020). *End-to-End Object Detection with Transformers*. arXiv: 2005.12872 [cs.CV]. URL: https://arxiv.org/abs/2005.12872.

Chai, Junyi et al. (2021). "Deep learning in computer vision: A critical review of emerging techniques and application scenarios". In: *Machine Learning with Applications* 6, p. 100134. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000670.

Chen, Liang-Chieh et al. (2016). *Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs*. arXiv: 1412.7062 [cs.CV]. URL: https://arxiv.org/abs/1412.7062.

Chen, Liang-Chieh et al. (2017a). *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. arXiv: 1606.00915 [cs.CV]. URL: https://arxiv.org/abs/1606.00915.

Chen, Liang-Chieh et al. (2017b). *Rethinking Atrous Convolution for Semantic Image Segmentation*. arXiv: 1706.05587 [cs.CV]. URL: https://arxiv.org/abs/1706.05587.

Chen, Liang-Chieh et al. (2018). *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. arXiv: 1802.02611 [cs.CV]. URL: https://arxiv.org/abs/1802.02611.

Chen, Yuantao et al. (2024a). "DNNAM: Image inpainting algorithm via deep neural networks and attention mechanism". In: *Applied Soft Computing* 154, p. 111392. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2024.111392. URL: https://www.sciencedirect.com/science/article/pii/S1568494624001662.

Chen, Yuantao et al. (2024b). "MFMAM: Image inpainting via multi-scale feature module with attention module". In: *Computer Vision and Image Understanding* 238, p. 103883. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2023.103883. URL: https://www.sciencedirect.com/science/article/pii/S1077314223002631.

Cheng, Bowen, Alexander G. Schwing, and Alexander Kirillov (2021). *Per-Pixel Classification is Not All You Need for Semantic Segmentation*. arXiv: 2107.06278 [cs.CV]. URL: https://arxiv.org/abs/2107.06278.

Cheng, Bowen et al. (2022). *Masked-attention Mask Transformer for Universal Image Segmentation*. arXiv: 2112.01527 [cs.CV]. URL: https://arxiv.org/abs/2112.01527.

Chi, Lu, Borui Jiang, and Yadong Mu (2020). "Fast Fourier Convolution". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 4479–4488. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/2fd5d41ec6cfab47e32164d5624269b1-Paper.pdf.

Cordts, Marius et al. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding*. arXiv: 1604.01685 [cs.CV]. URL: https://arxiv.org/abs/1604.01685.

Deng, Jia et al. (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

Dice, Lee R. (1945). "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26.3, pp. 297–302. ISSN: 00129658, 19399170. URL: http://www.jstor.org/stable/1932409 (visited on 2025-09-26).

Dosovitskiy, Alexey et al. (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

Esser, Patrick, Robin Rombach, and Björn Ommer (2021). *Taming Transformers for High-Resolution Image Synthesis*. arXiv: 2012.09841 [cs.CV]. URL: https://arxiv.org/abs/2012.09841.

Everingham, Mark et al. (2010). *The PASCAL Visual Object Classes (VOC) Challenge*. arXiv: 1001.0406 [cs.CV]. URL: https://arxiv.org/abs/1001.0406.

Everingham, Mark et al. (2015). "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1, pp. 98–136. DOI: 10.1007/s11263-014-0733-5. URL: https://doi.org/10.1007/s11263-014-0733-5.

Girshick, Ross (2015). *Fast R-CNN*. arXiv: 1504.08083 [cs.CV]. URL: https://arxiv.org/abs/1504.08083.

Girshick, Ross et al. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. arXiv: 1311.2524 [cs.CV]. URL: https://arxiv.org/abs/1311.2524.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

Gulrajani, Ishaan et al. (2017). *Improved Training of Wasserstein GANs*. arXiv: 1704.00028 [cs.LG]. URL: https://arxiv.org/abs/1704.00028.

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385.

He, Kaiming et al. (2018). *Mask R-CNN*. arXiv: 1703.06870 [cs.CV]. URL: https://arxiv.org/abs/1703.06870.

Heusel, Martin et al. (2018). *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. arXiv: 1706.08500 [cs.LG]. URL: https://arxiv.org/abs/1706.08500.

Hevner, Alan et al. (Mar. 2004). "Design Science in Information Systems Research". In: *Management Information Systems Quarterly* 28, pp. 75–.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). *Denoising Diffusion Probabilistic Models*. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

Howard, Andrew G. et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv: 1704.04861 [cs.CV]. URL: https://arxiv.org/abs/1704.04861.

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa (July 2017). "Globally and locally consistent image completion". In: *ACM Trans. Graph.* 36.4. ISSN: 0730-0301. DOI: 10.1145/3072959.3073659. URL: https://doi.org/10.1145/3072959.3073659.

Jaccard, Paul (Jan. 1901). "Etude de la distribution florale dans une portion des Alpes et du Jura". In: *Bulletin de la Societe Vaudoise des Sciences Naturelles* 37, pp. 547–579. DOI: 10.5169/seals-266450.

Jiang, Longtao et al. (2025). *SmartEraser: Remove Anything from Images using Masked-Region Guidance*. arXiv: 2501.08279 [cs.CV]. URL: https://arxiv.org/abs/2501.08279.

Karras, Tero, Samuli Laine, and Timo Aila (2019). *A Style-Based Generator Architecture for Generative Adversarial Networks*. arXiv: 1812.04948 [cs.NE]. URL: https://arxiv.org/abs/1812.04948.

Karras, Tero et al. (2018). *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. arXiv: 1710.10196 [cs.NE]. URL: https://arxiv.org/abs/1710.10196.

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

Kirillov, Alexander et al. (2019a). *Panoptic Feature Pyramid Networks*. arXiv: 1901.02446 [cs.CV]. URL: https://arxiv.org/abs/1901.02446.

Kirillov, Alexander et al. (2019b). *Panoptic Segmentation*. arXiv: 1801.00868 [cs.CV]. URL: https://arxiv.org/abs/1801.00868.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (May 2017). "ImageNet classification with deep convolutional neural networks". In: *Commun. ACM* 60.6, pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

Lauriola, Ivano, Alberto Lavelli, and Fabio Aiolli (2022). "An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools". In: *Neurocomputing* 470, pp. 443–456. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.05.103. URL: https://www.sciencedirect.com/science/article/pii/S0925231221010997.

Lin, Tsung-Yi et al. (2015). *Microsoft COCO: Common Objects in Context*. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

Lin, Tsung-Yi et al. (2017). *Feature Pyramid Networks for Object Detection*. arXiv: 1612.03144 [cs.CV]. URL: https://arxiv.org/abs/1612.03144.

Lin, Tsung-Yi et al. (2018). *Focal Loss for Dense Object Detection*. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/abs/1708.02002.

Liu, Guilin et al. (2018). *Image Inpainting for Irregular Holes Using Partial Convolutions*. arXiv: 1804.07723 [cs.CV]. URL: https://arxiv.org/abs/1804.07723.

Liu, Ziwei et al. (Dec. 2015). "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). *Fully Convolutional Networks for Semantic Segmentation*. arXiv: 1411.4038 [cs.CV]. URL: https://arxiv.org/abs/1411.4038.

Loshchilov, Ilya and Frank Hutter (2019). *Decoupled Weight Decay Regularization*. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

Lugmayr, Andreas et al. (2022). *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*. arXiv: 2201.09865 [cs.CV]. URL: https://arxiv.org/abs/2201.09865.

Minaee, Shervin et al. (2021). "Image segmentation using deep learning: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 44.7, pp. 3523–3542.

Mirza, Mehdi and Simon Osindero (2014). *Conditional Generative Adversarial Nets*. arXiv: 1411.1784 [cs.LG]. URL: https://arxiv.org/abs/1411.1784.

Paszke, Adam et al. (2016). *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation*. arXiv: 1606.02147 [cs.CV]. URL: https://arxiv.org/abs/1606.02147.

Pathak, Deepak et al. (2016). *Context Encoders: Feature Learning by Inpainting*. arXiv: 1604.07379 [cs.CV]. URL: https://arxiv.org/abs/1604.07379.

Peffers, Ken et al. (Jan. 2007). "A design science research methodology for information systems research". In: *Journal of Management Information Systems* 24, pp. 45–77.

Quan, Weize et al. (2024). *Deep Learning-based Image and Video Inpainting: A Survey*. arXiv: 2401.03395 [cs.CV]. URL: https://arxiv.org/abs/2401.03395.

Radford, Alec, Luke Metz, and Soumith Chintala (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. arXiv: 1511.06434 [cs.LG]. URL: https://arxiv.org/abs/1511.06434.

Ren, Shaoqing et al. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497.

Ribeiro, Tiago (2024). *InPainTor: Context-Aware Segmentation and Inpainting in Real-Time*. GitHub repository. URL: https://github.com/ipleiria-ciic/in-pain-tor.

Robbins, Herbert and Sutton Monro (1951). "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3, pp. 400–407. ISSN: 00034851. URL: http://www.jstor.org/stable/2236626 (visited on 2025-09-27).

Rombach, Robin et al. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models*. arXiv: 2112.10752 [cs.CV]. URL: https://arxiv.org/abs/2112.10752.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

Russakovsky, Olga et al. (2015). *ImageNet Large Scale Visual Recognition Challenge*. arXiv: 1409.0575 [cs.CV]. URL: https://arxiv.org/abs/1409.0575.

Sagong, Min-cheol (2022). *RORD: A Real-world Object Removal Dataset*. GitHub repository. URL: https://github.com/Forty-lock/RORD.

Sagong, Min-Cheol et al. (2022). "RORD: A Real-world Object Removal Dataset". In: *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press. URL: https://bmvc2022.mpi-inf.mpg.de/0542.pdf.

Sagong, Min-cheol et al. (2019). "PEPSI : Fast Image Inpainting With Parallel Decoding Network". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11352–11360. DOI: 10.1109/CVPR.2019.01162.

Salehi, Seyed Sadegh Mohseni, Deniz Erdogmus, and Ali Gholipour (2017). *Tversky loss function for image segmentation using 3D fully convolutional deep networks*. arXiv: 1706.05721 [cs.CV]. URL: https://arxiv.org/abs/1706.05721.

Shin, Yong-Goo et al. (Jan. 2021). "PEPSI++: Fast and Lightweight Network for Image Inpainting". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1, pp. 252–265. ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.2978501. URL: http://dx.doi.org/10.1109/TNNLS.2020.2978501.

Silva, Alexandre et al. (2025). "Artificial intelligence applied to the stone manufacturing industry: A systematic literature review". In: *Computers and Electrical Engineering* 128. JCR Impact Factor: 4.9 (2024); CiteScore: 10.7 (2024); SJR Impact Factor: 1.053 (2024); SJR Category Rank (2024): Computer Science (miscellaneous): Q1 | Control and Systems Engineering: Q1 | Electrical and Electronic Engineering: Q1, p. 110702. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2025.110702. URL: https://www.sciencedirect.com/science/article/pii/S0045790625006457.

Silva, David (2018). *PyTorch-ENet*. GitHub repository. URL: https://github.com/davidtvs/PyTorch-ENet.

Simonyan, Karen and Andrew Zisserman (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

Song, Shuran, Samuel P. Lichtenberg, and Jianxiong Xiao (2015). "SUN RGB-D: A RGB-D scene understanding benchmark suite". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.

Sudre, Carole H. et al. (2017). "Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations". In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer International Publishing, pp. 240–248. ISBN: 9783319675589. DOI: 10.1007/978-3-319-67558-9_28. URL: http://dx.doi.org/10.1007/978-3-319-67558-9_28.

Suvorov, Roman et al. (2021). *Resolution-robust Large Mask Inpainting with Fourier Convolutions*. arXiv: 2109.07161 [cs.CV]. URL: https://arxiv.org/abs/2109.07161.

Szegedy, Christian et al. (2014). *Going Deeper with Convolutions*. arXiv: 1409.4842 [cs.CV]. URL: https://arxiv.org/abs/1409.4842.

Vaswani, Ashish et al. (2023). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

Xiao, Jianxiong et al. (2010). "SUN database: Large-scale scene recognition from abbey to zoo". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492. DOI: 10.1109/CVPR.2010.5539970.

Xiao, Jianxiong et al. (2016). "SUN Database: Exploring a Large Collection of Scene Categories". In: *International Journal of Computer Vision* 119.1, pp. 3–22. DOI: 10.1007/s11263-014-0748-y. URL: https://doi.org/10.1007/s11263-014-0748-y.

Yu, Changqian et al. (2018). *BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation*. arXiv: 1808.00897 [cs.CV]. URL: https://arxiv.org/abs/1808.00897.

Yu, Jiahui et al. (2018). *Generative Image Inpainting with Contextual Attention*. arXiv: 1801.07892 [cs.CV]. URL: https://arxiv.org/abs/1801.07892.

Yu, Jiahui et al. (2019). *Free-Form Image Inpainting with Gated Convolution*. arXiv: 1806.03589 [cs.CV]. URL: https://arxiv.org/abs/1806.03589.

Yu, Ying et al. (2023). "Techniques and Challenges of Image Segmentation: A Review". In: *Electronics* 12.5. ISSN: 2079-9292. DOI: 10.3390/electronics12051199. URL: https://www.mdpi.com/2079-9292/12/5/1199.

Zhang, Richard et al. (2018). *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. arXiv: 1801.03924 [cs.CV]. URL: https://arxiv.org/abs/1801.03924.

Zhao, Shengyu et al. (2021). *Large Scale Image Completion via Co-Modulated Generative Adversarial Networks*. arXiv: 2103.10428 [cs.CV]. URL: https://arxiv.org/abs/2103.10428.

Zhou, Bolei et al. (2017). "Places: A 10 million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, Bolei et al. (2018). *Semantic Understanding of Scenes through the ADE20K Dataset*. arXiv: 1608.05442 [cs.CV]. URL: https://arxiv.org/abs/1608.05442.

Zhuang, Junhao et al. (2024). *A Task is Worth One Word: Learning with Task Prompts for High-Quality Versatile Image Inpainting*. arXiv: 2312.03594 [cs.CV]. URL: https://arxiv.org/abs/2312.03594.

This page intentionally left blank.