Generative Models for Context Dependent Urban Planning

Master’s Thesis in Complex Adaptive Systems

Wojciech Napieralski

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's Thesis 2025

Generative Models for Context Dependent Urban Planning

Wojciech Napieralski

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2025

Generative Models for Context Dependent Urban Planning

Wojciech Napieralski

Supervisor: Alexander Radne, Parametric Solutions
Examiner: Mats Granath, Department of Physics

Master's Thesis 2025
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Density guided building footprint design suggestions generated with a modified and fine-tuned image inpainting network.

Typeset in LaTeX
Gothenburg, Sweden 2025

Generative Models for Context Dependent Urban Planning
Wojciech Napieralski
Department of Physics
Chalmers University of Technology

Abstract

Building footprints represent the total area of coverage of a physical building. Footprints can be used to give an overview of a planned developed area in the early stages of urban planning. This thesis investigates the possibility of training a generative AI model to generate building footprints in a designated area based on surrounding, already existing building footprints. Such a generative model would be a useful tool for architects in the early stages of urban planning, as it would allow the rapid generation of footprint suggestions in an area designated for development. Two image inpainting networks trained for general image reconstruction were fine-tuned using a dataset of building footprints to improve on the task of footprint generation. One of the networks was also modified and trained to be able to accept a desired density of footprints in the generated area as an additional input. Two different masking algorithms were used during training and evaluation: A simple square approach and a more sophisticated algorithm that finds and masks city blocks. FID and LPIPS were used to evaluate and compare the trained models. It was shown that image inpainting networks form a good basis for context dependent footprint generation and that fine-tuning improves performance on this task. Furthermore, it was demonstrated that an image inpainting network can be modified to accept and adhere to density requirements providing a proof-of-concept for other types of user guidance.

Acknowledgements

I would like to thank Parametric Solutions for providing me with the resources needed to conduct this thesis and for an exceptionally pleasant workspace environment. In particular, I would like to thank my supervisor, Alexander Radne, for his guidance, interest, and support. I would also like to thank my examiner, Mats Granath, for his guidance. Finally, I would like to thank all my study mates—Joel and Albin in particular—for making my time studying MPCAS the most rewarding and humorous years of my Chalmers education.

Wojciech Napieralski, Gothenburg, Januari 2025

Contents

1 Introduction **1**
1.1 Introduction 1

2 Theory **3**
2.1 Machine Learning Applications in Image Processing 3
2.1.1 Raster Graphics 3
2.1.2 Convolution Neural Networks 3
2.2 Image Inpainting 5
2.2.1 Autoencoders 5
2.2.2 Masking 6
2.2.3 Generative Adversarial Networks 7
2.2.4 Training Inpainting Networks 8
2.3 Transfer Learning 8
2.4 Deepfillv2 9
2.4.1 Gated Convolution 9
2.4.2 Spectral-Normalized Markovian Discriminator (SN-PatchGAN) 10
2.4.3 GAN Hinge-Loss 11
2.4.4 Network Summary 11
2.5 LaMa 11
2.5.1 Fast Fourier Convolution 12
2.5.2 ResNet 12
2.5.3 Perceptual Loss 13
2.5.4 Network Summary 13
2.6 Assessing the Performance of Image Inpainting Models 14
2.6.1 Fréchet Inception Distance 14
2.6.2 LPIPS 15

3 Methods **17**
3.1 Data 17
3.2 Masking 18
3.3 Transfer Learning 19
3.4 User Guided Density 20
3.5 Evaluation 21

4 Results **23**
4.1 Quality analysis 23
4.2 Further Evaluation of User Guided Density Model 25
4.3 Block Mask Model 27

5 Discussion **29**
5.1 Baseline and Fine-Tuned Models 29
5.2 Block Masks 30
5.3 User Guided Models 30
5.4 Evaluation 31
5.5 Method Suitability 31

6 Conclusion **33**

A Appendix 1 **I**

1 Introduction

1.1 Introduction

Although Artificial Intelligence (AI), and machine learning (ML) in particular, has been in commercial use for decades, the recent advancements in generative AI (GenAI) technologies have resulted in increasing commercial interest.

Undoubtedly, many companies choose to invest in GenAI because automating parts or entire generative tasks could lead to significant financial and time savings. Developing a GenAI model from scratch can prove to be an expensive endeavor when considering costs such as research, data acquisition, model training, and deployment. Therefore, the possibility of developing in-house models GenAI can prove to be difficult for smaller companies. Instead of creating new ML models, one can adapt existing models to perform the desired task. Such an approach generally requires less data [1] and training [2] and is referred to as transfer learning.

This thesis will focus on a particular type of generative task called image inpainting. Inpainting aims to infer missing parts of an image based on the information given in the existing parts of an image. Inpainting can be used in a variety of tasks including image restoration, removing obstructions, and image completion. By considering both the local aspects around the missing region and the global context of the entire image the inpainted region will fit with the structure of the image and be context dependent.

In particular, the aim is to use inpainting in order to generate building footprint suggestions for a particular urban area using the surrounding area to guide the generative process. Footprints in the context of urban planning are polygons used to represent the total area covered by a physical building. Furthermore, transfer learning will be used in order to save on computational resources. This in turn provides an opportunity to provide insight to the topic of transfer learning on inpainting networks, which has not been thoroughly explored in previous works. Finally, this thesis aims to investigate if other types of information can be used as inputs in order to further guide the generative process of the networks.

This thesis primarily investigates GenAI models capable of generating footprint suggestions in a designated area by considering the surrounding architecture. This kind of AI model could be used to guide an architect in early decision-making and accelerate the process of early urban planning. However urban planning is a complex task.

Even in the early stages of planning many different factors need to be considered. These include but are not limited to: Parking, daylight access, noise pollution, and air quality [3]. Geographical information about these factors is often available and used to guide the architect's decision-making. Although not investigated, the same methods developed in this thesis could be utilized to incorporate this information as well. Finally, a case could be made that such information is already indirectly encoded in the geometric representations of urban structures, as these were created by architects with the aforementioned factors in mind. Consequently, it could be indirectly utilized during inference by a neural network focusing primarily on geometric data. Still, there is value in allowing the user to specify parameters such as density or parking availability for the network to consider, giving the user more creative control.

Two inpainting models form the basis for the work presented in this report. These are Deepfillv2 [4] and LaMA [5]. Both models will be described in upcoming chapters.

GenAI for urban structure suggestions has been explored in a limited fashion in earlier works. These include training models to produce urban design schemes corresponding to certain parts of a city [6] and generating context dependent urban blocks, using networks not specifically designed for inpainting, with some success [7].

2
Theory

This chapter aims to provide the technical foundation necessary to understand this thesis report. It assumes that the reader has an understanding of the essential topics of machine learning, in particular supervised learning. Fundamental topics will be covered focusing on conceptual understanding rather than mathematics. Advanced topics relevant to the thesis will be covered in more technical detail.

2.1 Machine Learning Applications in Image Processing

ML is commonly applied to image data. This could include tasks such as classification, generation, and inpainting. ML on image data is challenging since the ML algorithm may need to understand the spatial dependencies between elements of the image in order to solve the required task.

2.1.1 Raster Graphics

Raster graphics are a way to represent 2-dimensional images in digital form [8]. A raster image is a 2-dimensional grid of pixels. Each pixel consists of a numerical value corresponding to a certain color. Raster images are therefore approximations that become more accurate as the number of pixels, or resolution, increases. Color images consist of pixels with three values called color channels. Each channel corresponds to Red, Blue, and Green (RGB) respectively. Some RGB formats have an additional alpha channel that corresponds to opacity. In the context of ML, raster images are represented as tensors with shape $H \times W \times C$ where H is pixel height, is W pixel width and C is number of channels. Consequently, as the resolution increases, image tensors experience a rapid increase of variables. Therefore, simply using deep neural networks with linear layers becomes problematic as each variable requires a connection to input layer resulting in large, inefficient networks.

2.1.2 Convolution Neural Networks

Convolutional Neural Networks (CNN) present a solution to the problem of high dimensionality mentioned previously. Any neural network utilizing one or more *convolutional layer* is commonly referred to as a CNN [9]. Convolutional layers apply a mathematical operation called convolution. A convolution in the context of ML consists of taking an input and a kernel and performing *cross-correlation* to obtain an output, as illustrated in Figure 2.1.

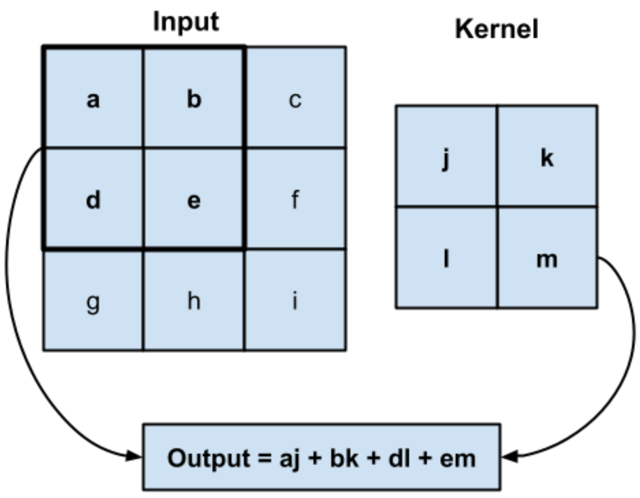


Figure 2.1: Illustration of cross-correlation between parts of an image and a kernel.

The kernel will "move" in a typewriter scheme across the entire input, meaning from left to right and top to bottom. The resulting feature map, see Figure 2.2, consists of cross-correlations between the kernel and different parts of the input. How much the kernel should "move" between each cross-correlation is referred to as stride.

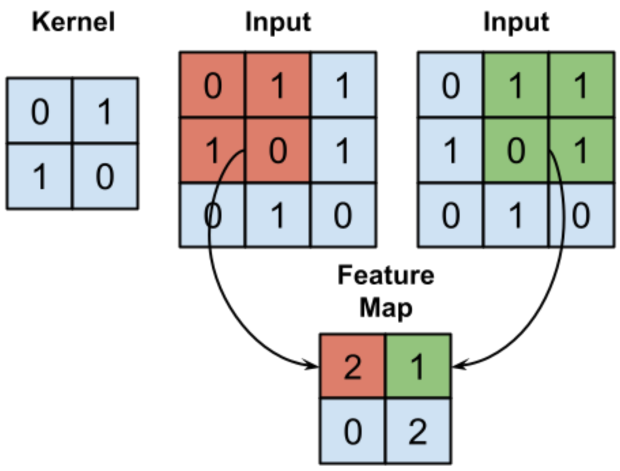


Figure 2.2: Illustration of convolution between an input and a kernel. In this example, the stride is set to 1 and the kernel will move one step between each cross correlation. Illustrated in red is the cross correlation between the kernel and the top-left part of the input. Illustrated in green is the cross correlation after the kernel has moved one step in the right direction. The second row of the feature map shows the result of cross-correlation after the kernel has moved down and along the lower part of the input.

This description explains a convolutional layer in its simplest form and is intended to give the reader a conceptual understanding of how a CNN operates. In practice, a convolutional layer consists of multiple kernels each producing a local map stored as a channel in the output. Convolutional layers are often stacked, with each subsequent layer applied to the feature maps produced by previous layers. While an output in a feature map resulting from an early layer may contain information from a small part of the input, outputs from feature maps deeper in the network will contain more global information.

It is important to note that kernels are essentially the weights of the network. During training, the kernels are optimized in order to give the network the capability to recognize certain features of the input image. An example would be a CNN capable of recognizing human faces. After training, kernels belonging to early convolutional layers would be optimized to look for local details such as corners of the eyes or parts of the mouth. Kernels belonging to deeper layers would in turn look for global features such as facial structure.

Neural networks that consist exclusively of convolutional layers are known as Fully Convolutional Networks (FCNs). Since the weights of an FCN are entirely determined by the parameters of the convolutional kernels, these networks can accept inputs of varying dimensions. As a result, FCNs are capable of generalizing to inputs of different sizes, even after being trained on inputs of a specific dimension.

2.2 Image Inpainting

Image inpainting aims to, given an incomplete image, restore the missing pixel features [10]. The output should be a high quality approximation of the original undamaged image. In the context of ML, the difficulty of such a task may vary depending on the situation at hand. An image with a clear structure and pattern, such as a photo of a chain-link fence with missing regions, may present a simpler challenge compared to an image with more varying features. In the case of the chain-link fence, an appropriate approximation would be more chain-link fencing with the same structure and color as the surrounding fence. However if the image is complex and varied, or the damaged parts are extensive, it might be hard to restore the image as many different approximations could be considered valid.

The objective of training an inpainting network is to obtain a generator capable of, given an input image masked with a binary mask of unknown pixels, generating an inpainted image with the unknown pixels predicted in a reasonable fashion. Inpainting generators are usually fully convolutional feed-forward networks that extract features from the surrounding undamaged pixels and then reconstruct the missing areas using the extracted information.

2.2.1 Autoencoders

An inpainting generator using convolutional layers consists of two parts, an encoder and a decoder. Such a pair is used in the context of an autoencoder. The task of the encoder is to extract the important features of the input into an "informative" lower dimension representation [11] henceforth referred to as a *latent space representation*. This is achieved by stacking multiple convolutional layers sequentially to down-sample the image. The task of the decoder is to up-sample or translate the latent space representation back to the original image by utilizing consecutive inverse convolutions. When training an autoencoder the image and the target output is usually the same. If trained successfully, the autoencoder learns to extract the most relevant features of the data in addition to learning how to reconstruct the data from latent space.

In the context of image inpainting, autoencoders are utilized in a similar manner but with one important distinction. The goal is now to extract the important features of an image with missing content into latent space while being able to reconstruct the latent space representation into an image where the missing content is predicted. The masked image is used as the input, while the original, unmasked image serves as the target, see Figure 2.3. This approach trains the autoencoder to predict the missing pixels based on the latent space representation of the unmasked pixels.

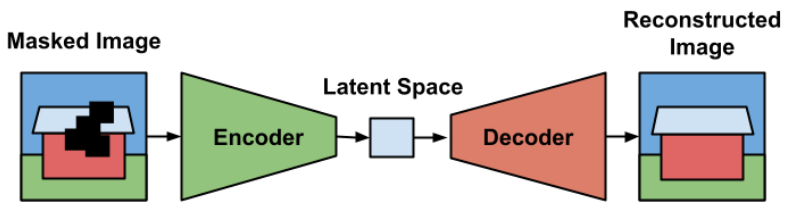


Figure 2.3: Illustration of the principles of an inpainting autoencoder.

2.2.2 Masking

As mentioned earlier, the input to an inpainting generator is a masked image. During training, masked images are created by sampling an unmasked image x from the training dataset. A binary mask m is generated by a masking algorithm as opposed to having fixed image-mask pairs in the dataset. A masked image tensor x' is obtained as the element-wise product between x and m. Additionally, the binary mask is concatenated to the masked image in the channel dimension. The end result is a tensor of dimension $H \cdot W \cdot (C + 1)$ where H,W,C denotes the height, width and number of channels of x and $\odot$ is element-wise multiplication:

$$x' = concatenate(x \odot m, m) \tag{2.1}$$

Masking approach has been shown to have significant impact on network performance [12, 4]. It is important to generate masks that both resemble the intended real use-case and are diverse to avoid over-fitting.

2.2.3 Generative Adversarial Networks

Originally proposed by Goodfellow et. al. in 2014 [13], Generative Adverisal Networks (GAN) are commonly utilized in deep learning generative tasks. In many cases, inpainting networks are trained as GANs. So far in this chapter, we have only discussed the parts of an inpainting network that allow it to learn and copy the ground truth from images. If the network is only judged on its capability to generate predictions that are as similar as possible to the ground truth, how can it learn to generalize to more complex inpainting tasks? As discussed earlier, an inpainting task become more challenging when a wider range of predictions could be considered suitable.

A GAN consists of two networks: A generator and a discriminator. In the case of inpainting, the generator is an inpainting autoencoder as described earlier while the discriminator is a classification network. The decoder's task is to classify an input image as either real or fake. During training, the generator and discriminator are pitted against each other: the generator takes masked images as input and predicts the missing pixels, while the discriminator receives either the generator's inpainted images or real, unaltered samples from the training data. By forcing the generator to compete against a discriminator, it is forced to generate predictions that are not only similar to the ground truth of the given data point but also true to the distribution of the entire dataset. In other words, the generator is forced to make realistic predictions in order to fool the discriminator.

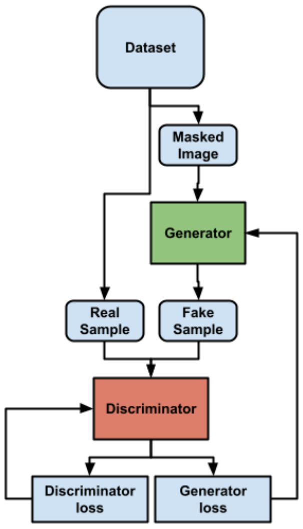


Figure 2.4: Schematic showcasing the pipeline of a typical GAN.

During training, adversarial loss consisting of generator loss and discriminator loss is calculated based on the discriminator's predictions. GAN adversarial loss can be formulated in multiple different ways but the principle remains the same: The discriminator is rewarded or penalized based on its ability to correctly classify real and generated samples. The generator, on the other hand, is rewarded if the discriminator incorrectly classifies a generated sample as real and penalized if the discriminator correctly classifies it as fake. The adversarial loss is then used to optimize the weights of the two networks to perform better against each other. The two networks are trained in parallel.

The generator and the discriminator are essentially playing a zero-sum game against each other, where the success of one is detrimental to the other. In theory, the performance of the two networks should stabilize in a saddle point, but in practice, balancing this game might be hard. If the discriminator performs too well it might overpower the generator. This in turn results in vanishing gradients during back-propagation for the generator[14]. Another common issue is mode collapse: This occurs when the generator learns to map a large amount of inputs to one single output capable of fooling the discriminator. This results in the generator with limited generative capabilities as it is stuck to outputting a limited set of predictions regardless of input.

2.2.4 Training Inpainting Networks

The fundamental building blocks of an inpainting network have now been described. To reiterate, inpainting networks commonly consist of a fully convolutional, autoencoder-based generator. In addition, during training a discriminator is utilized to form a GAN. The final loss function for a generator is often the sum of multiple loss functions, in particular reconstruction loss and generator adversarial loss. The reconstruction loss is based on a comparison between the prediction and the unmasked input while the generator adversarial loss is based on the generator's ability to fool the discriminator. By utilizing both metrics, the generator learns not only to infer by looking at the missing pixels but also by considering the distribution of the entire dataset. This results in a greater creative capability as predictions deviating from ground truth are not as heavily penalized, given that these predictions are considered realistic by the discriminator.

In practice, there are many possible variations to an inpainting network regarding all aspects described above. Inpainting networks might use different network layers, loss functions, and masking approaches. They do not necessarily need to be autoencoder based, GANs or only utilize convolutional layers[5]. The inpainting networks used in this thesis do use the concepts described earlier and will be described in more detail later in this chapter.

2.3 Transfer Learning

Transfer learning refers to leveraging the knowledge a machine learning model has gained from training on one task to perform a different, but similar, task. Consider a domain $\mathcal{D} = \{\chi, P(X)\}$ consisting of a feature space χ and probability distribution $P(X)$ where $X = \{x_1, ..., x_n\} \in \chi$. A task can be described as $\mathcal{T} = \{\mathcal{Y}, f(x)\}$ where $\mathcal{Y}$ is a label space and f is a predictive function that maps $\chi \to \mathcal{Y}$. The formal definition of transfer learning then becomes: Given a target task $\mathcal{T}_t$ on target domain $\mathcal{D}_t$ and source task $\mathcal{T}_s$ on source domain $\mathcal{D}_s$, where $\mathcal{D}_s \neq \mathcal{D}_t$ or $\mathcal{T}_s \neq \mathcal{T}_t$, transfer learning aims to improve the performance of the target predictive function $f_t(\cdot)$ by transferring knowledge gained from $\mathcal{D}_s$ and $\mathcal{T}_s$ [15].

In this thesis, two approaches to transfer learning are used. The first one could be described as a "naive" approach, where inpainting models learned on a very large dataset are directly applied on a different, much smaller dataset. This approach could be extremely effective, since no further training would be required but assumes that $\mathcal{D}_s$ and $\mathcal{D}_t$ are similar. If the shift between the two domains is too large, the knowledge gained from $\mathcal{D}_s$ might not be applicable on $\mathcal{D}_t$.

The second approach is known as fine-tuning. Here the same models as in the naive approach are used, but additional supervised training on data from $\mathcal{D}_t$ is used to improve inpainting performance. This approach assumes that knowledge learned from $\mathcal{D}_s$ and $\mathcal{T}_s$ is relevant to $\mathcal{T}_t$, but that additional knowledge from $\mathcal{D}_t$ might boost performance further. A network trained on data from $\mathcal{D}_s$ is used to initialize the weights and the network is trained to perform $\mathcal{T}_t$. This approach is useful when the dataset for the new task is to small for the network to learn sufficiently. By fine-tuning a network trained on a larger dataset, it has been shown that for computer vision tasks, significantly less data was required to train the model on a new task [2].

2.4 Deepfillv2

Deepfillv2 [4] is one of the two inpainting networks utilized in this thesis. Although considered state of the art when introduced in 2019, it has since been surmounted by subsequent models [12, 16, 17]. Nevertheless, Deepfillv2 is a capable model considering its relatively small size of roughly 6 million parameters. Deepfillv2 is a fully convolutional, autoencoder-based inpainting network trained as a GAN but seeks to address some of the issues that arrive from using standard CNNs and GANs.

Inpainting networks consisting of standard convolutional layers often generate inpainted images with artifacts, distortions, and blurry regions. It is theorized that these issues arrive because CNNs struggle to infer the missing pixels from distant parts of the image. Deepfillv2 presents a solution to this problem by utilizing *contextual attention layers* to capture these long-range dependencies [18]. Contextual attention introduces an attention mechanism that allows the network to learn where to copy features from an image globally in order to generate the missing region. Although interesting, this thesis will not focus on contextual attention in further detail. Furthermore, *gated convolutions* [4] are utilized to address some of the issues stemming from applying standard CNNs on irregular mask shapes.

2.4.1 Gated Convolution

A standard CNN treats all pixels in an image equally, meaning it does not differentiate between valid (unmasked) and invalid (masked) pixels in the input. This results in the network propagating unimportant information, such as the ones or zeros from the mask region or synthesized pixels in deeper layers, throughout the network. As a consequence, the network's predictive capabilities suffer, as the information from the masked region impairs its ability to infer from the unmasked region. *Gated Convolution* was proposed as a solution to this issue [4].

Gated convolution layers consist of two types of kernels, feature kernels W_f, that operate in the same manners as kernels in an ordinary CNN, and gating kernels W_g. The objective of the gating kernels is to learn a "soft" mask from the data. In this way, the network can understand better which features to consider and which to ignore, even in deeper layers. Gated convolution is formulated as [4]:

$$\begin{aligned} Gating &= \sum\sum W_g \cdot I \\ Feature &= \sum\sum W_f \cdot I \\ O = \phi(&Feature) \odot \sigma(Gating) \end{aligned} \tag{2.2}$$

where I is the input to the gated convolution layer, O is the output of the gated convolution, σ is a sigmoid function and ϕ is any activation function as used in standard convolution. $\sigma(Gating)$ will consist of elements ranging from zero to one and can be thought as a soft mask multiplied pixel-wise with $\phi(Feature)$ in order to assign importance to different regions of the input. This is learned during training. The gating mechanism helps the network to differentiate relevant regions from irrelevant regions regardless of mask shape (as long as the network has been trained on diverse masks).

2.4.2 Spectral-Normalized Markovian Discriminator (SN-PatchGAN)

Some earlier inpainting networks were trained on a consistent mask shape, commonly a square rectangle. To improve performance two discriminator networks were utilized, one global that considered the entire image, and one local that only discriminates the inpainted region[19]. This method has been proven to be unsuitable when training with diverse masks where multiple, unconnected, masked regions might exist [4]. To overcome the above-mentioned problem, Deepfillv2 utilizes a *Markovian Discriminator* [20] that functions both as a local and global GAN. The Markovian discriminator consists of a convolutional neural network that maps the input to latent space with dimensions W, H, C, that correspond to width, height and number of channels of the latent space representation respectively. Each feature of the latent space representation is classified as real or fake separately. In essence, a Markovian discriminator consists of $W \times H \times C$ number of GANs applied locally on different areas (and different semantic representations in the channel dimension) of the input image [4]. Furthermore, since each feature in latent space contains global information as well as a result of sequential convolutions, the Markovian discriminator also functions as a global GAN.

To stabilize training, *spectral normalization* [21] is used to regularize the discriminator's weights. The objective of spectral normalization is to control the Lipschitz constant of the discriminator by constraining the spectral norm of the weights W_d^l of each layer l in the discriminator. More specifically, the discriminator is regularized to have weights that satisfy the constraint $\sigma_{max}(\hat{W_d^l}) = 1$, where σ_{max} is the spectral norm, and $\hat{W}_d^l$ is the the spectral normalized weight matrix. $\hat{W}_d^l$ is given as:

$$\hat{W}_d^l = \frac{W_d^l}{\sigma_{max}(W_d^l)} \tag{2.3}$$

This constraint ensures that the weight matrices of the discriminator are Lipschitz continuous and 1-Lipschitz. Spectral normalization has been shown to improve GAN training by ensuring that the discriminator function is smooth and does not change too rapidly, promoting more stable and reliable training dynamics [21].

2.4.3 GAN Hinge-Loss

Deepfillv2 utilities GAN hinge-loss as adversarial loss. Hinge-loss consists of discriminator loss:

$$\begin{aligned} L_D = \frac{1}{2}\Big(&\mathbb{E}_x\left[\max(0, 1 - D^{SN}(x))\right] \\ &+ \mathbb{E}_z\left[\max(0, 1 + D^{SN}(G(z)))\right]\Big) \end{aligned} \tag{2.4}$$

where D^{SN} is the spectral normalized discriminator, G is the generator, x is a ground truth unmasked image from the dataset and z is a incomplete masked image, and generator loss:

$$L_G = -\mathbb{E}_z\left[D(G(z))\right] \tag{2.5}$$

To minimize its loss, the discriminator needs to assign a value ≥ 1 to real images x and ≤ -1 to fake images $G(z)$. Similarly, per equation 2.5, L_G becomes smaller if D^{SN} misclassifies $G(z)$ by assigning values > -1 and vice versa.

2.4.4 Network Summary

Deepfillv2 consists of two stages. The first stage is an autoencoder consisting of stacked gated convolutional layers that produce a rough inpainted image. This image is then inserted into a second network that refines the image. The refinement network consists of two parallel encoders that feed into a single decoder. The first encoder consists of stacked gated convolutional similar to the first stage while the second decoder has an attention module consisting of contextual attention layers in order to attend to longe range dependencies. The original paper [4] provides an excellent illustration of the network architecture. During training, the final loss function of the inpainting network is composed of pixel-wise mean-square error (MAE) calculated between the inpainted images and unmasked ground truth images combined with adversial loss from the SN-PatchGAN.

2.5 LaMa

Introduced in 2021, LaMa [12] (not to be confused with Meta's large language model called Llama) is a considerably more capable model compared to Deepfillv2, but significantly larger at roughly 51 million trainable parameters. Considering this, LaMa is still 3-4 times smaller than models with comparative performance [12]. LaMa is fully convolutional, being able to generalize to much higher resolutions than during training.

2.5.1 Fast Fourier Convolution

Fast Fourier convolution (FFC) [22] allow the neural network to gain a global understanding of the image in earlier layers compared to standard convolutional layers. As mentioned earlier, standard CNNs require stacking multiple layers in order for the feature maps to contain global information. This means that only later parts of the network will have the global understanding needed to be capable of high-quality inpainting. By enabling the network to gain global information earlier, it becomes more effective [12].

FFC works by splitting the input into two different branches, a local branch that uses standard convolutions, and a global branch that uses fast Fourier transforms (FFT) [23]. FFT is an algorithm that computes the discrete Fourier transform, or its inverse, from a given sequence. In FFC, FFT is used to map an input image or feature map from its original spacial domain into a frequency domain. A convolutional block is applied to the frequency domain and then inverse FFT is used to convert back to the spacial domain [12].

Consider an input x of dimension $H \times W \times C$ where H, W, C denotes the height, width, and number of channels. FCC splits x along the channel dimensions as $x = \{x^l, x^g\}$, where x^l is the local part and x^g is the global part. x^l and x^g are processed separately by the local and global branches respectively to produce a local output part y^l and a global output part y^g. Finally, y^l and y^g are concatenated to form the final output. This system allows FFC to consider local features in the same manner as a standard convolutional layer, while also gaining insight into periodic global features. It has been shown FCC excels in capturing periodic structures in images, such as windows, fences and other man-made structures [22].

2.5.2 ResNet

Residual Networks (ResNet) [24] were originally developed for image classification to address challenges arising from increasingly deeper neural networks. Since then, ResNet has formed the foundation for many subsequent computer vision neural networks.

The fundamental building block of ResNet is the residual block. Residual blocks introduce residual connections between the input x and the output y of a residual block as $y = F(x) + x$, where $F(x)$ is the residual block consisting of any number or types of layers, see Figure 2.5. This formulation tasks the network to learn the residuals, meaning the difference, between the input and the output rather than the complete mapping. This has the potential to improve learning in deep networks by presenting a solution to the so-called degradation problem. In theory, as networks become deeper they become better at generalizing, but in practice, deeper networks might struggle to learn to map the input to its identity $F(x) = x$. If the desired output of a layer is close to its identity, the network struggles resulting in reduced performance. The residual connections ensure that identity can be recreated

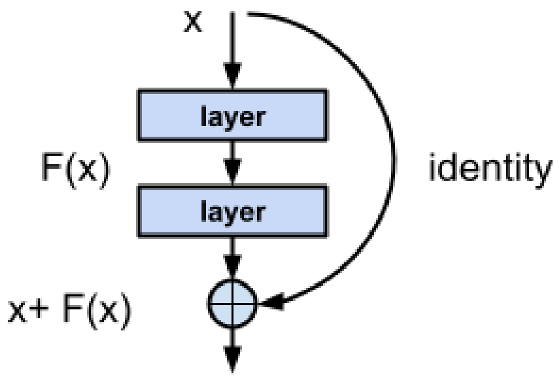


Figure 2.5: Schematic of a residual block.

by setting $F(x) = 0$. Intuitively, it allows the network to pass an input through redundant parts of the network without distorting the input unnecessarily.

2.5.3 Perceptual Loss

Pixel-wise reconstruction loss, such as used in Deepfillv2, requires the network to reconstruct the ground truth in order to minimize loss. As mentioned earlier, inpainting is an ambiguous task where ground truth is not necessarily easy to infer. In such cases, pixel-wise reconstruction loss in not a suitable metric, since inferring the precise ground truth might be impossible or only partly relevant.

An alternative to pixel-wise reconstruction loss is *perceptual loss* [12], which evaluates the distance between latent space representations of inpainted image and ground truth. This is achieved by using a pre-trained CNN for feature extraction. The choice of a pre-trained network is important in order to obtain relevant latent space representations since this network is not optimized during training. Perceptual loss in LaMa is formulated as a mean squared error (MSE) loss between latent space representations obtained from a ResNet-based feature extractor.

2.5.4 Network Summary

LaMa consists of a single autoencoder composed of convolutional down-sampling blocks, residual blocks with FFC layers instead of standard convolutional layers, and up-sampling blocks. It is trained as a GAN with patch-based adversarial loss similar, in concept, to the loss used by Deepfillv2. In addition, a *discriminator feature matching loss* [25] is used to improve training stability. Feature matching loss is computed by measuring the difference between features extracted from different layers of the discriminator for real and fake images. The final loss function is formulated as:

$$L = \kappa L_{adv} + \alpha L_{PL} + \beta L_{DFM} + \gamma R_1 \tag{2.6}$$

where L_{adv} is adversial loss, L_{PL} is perceptual reconstruction loss, L_{DFM} is discriminator feature matching loss and $R_1 = E_x \|\nabla D_\xi(x)\|^2$ is gradient penalty. κ, α, β and γ are hyperparameters.

During training, large mask generation, as much as 60 % of the image, is emphasized. LaMa is capable of inpainting large areas, in fact, training on large masks showed improved performance on smaller masks as well [12].

2.6 Assessing the Performance of Image Inpainting Models

Evaluating generative models is challenging because the inherent ambiguity of creating something new introduces a degree of subjectivity during evaluation. The simplest form of evaluation is visual evaluation. In the context of image inpainting, this consists of looking at an inpainted image to determine the quality of the generated content. Things to consider could be sharpness, lack of artifacts, and whether the inpainting blends in and fits with the context of the image at large. Such an approach is expensive as it requires the involvement of human evaluators who must manually inspect each image to assess its quality. This process is time-consuming, subjective, and does not scale well when evaluating large datasets or comparing multiple models. To overcome this, researchers have developed data-driven, objective methods to evaluate generated images.

2.6.1 Fréchet Inception Distance

Fréchet Inception Distance (FID) [26] is the most established metric when evaluating inpainting models. Similar to perceptual loss, FID compares the latent space representations of real and fake images obtained from a pre-trained feature extractor. Latent space representations of a set of real images r and a set of generated images g are obtained and FID is then calculated from the distribution of the two sets as:

$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{\frac{1}{2}}) \tag{2.7}$$

where μ_r, μ_g are the mean values of r and g respectively and Σ_r, Σ_g are the covariance matrices. Essentially, FID compares how similar the distribution of g is compared to r. The closer the distributions of the two sets are, the lower the FID score is, indicating a better model. The motivation behind this lies once again in the ambiguity of the task. Comparing pixel-wise is not relevant, as the goal is not to create a network that can reconstruct an image pixel perfectly to ground truth. Instead, FID compares how the essential features of generated images compare to the essential features of ground truth. As in perceptual loss, the choice of the feature extraction network needs to be considered, as it determines what those essential features are. Comparing FID scores between models trained on different datasets is not relevant, as FID not only reflects on the generative networks' capabilities but also the feature extractors ability to extract relevant features. A feature extractor trained on a dataset that differs greatly from the dataset the evaluated model is trained on might struggle with extracting relevant features.

2.6.2 LPIPS

An alternative to FID is Learned Perceptual Image Patch Similarity (LPIPS) [27]. LPIPS compares features extracted from different layers of a pre-trained feature extractor for real and generated images. Consider a real image x_r and a generated image x_g. Features are extracted from L different layers of the pre-trained network. Features extracted from layer l are denoted as y_r^l and y_g^l with dimensions $H_l \times W_l \times C_l$. LPIPS is calculated as:

$$LPIPS = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \| w_l \odot (y_{r,hw} - y_{g,hw}) \|_2^2 \tag{2.8}$$

where w_l is a learned weight that assigns different importance to features extracted from different layers. w_l is obtained by training LPIPS on a dataset of fake and real images labeled by humans. Intuitively, LPIPS evaluates images similarly to how a human would by comparing different aspects of the image such as local and global features.

3
Methods

This chapter will describe the methods used in this thesis. It will cover the format and acquisition of the data used for fine-tuning and evaluation of inpainting models on the task of generating footprint suggestions. The process of transfer learning and evaluation will be outlined as well. Finally, a modified Deepfillv2 model capable of user guided inpainting will be described.

3.1 Data

To form a basis for transfer learning and evaluation, a dataset of semantic map representations of city areas was collected. The images were obtained from OpenStreetMap[28] via Mapbox Static Image API [29]. Each data point is a 512x512-pixel PNG image representing an area of approximately 600x600 meters, with slight variations depending on the longitudinal position of the extracted region.

Each data point is color-coded using five colors, corresponding to the following categories: building footprint, road, pavement, waterfront, and miscellaneous land use. In the context of this thesis, miscellaneous land use includes areas that do not fall into the other categories, such as parks, forests, and fields. See Table 3.1 for color representations.

Category	**Color (RGB)**	**Color**
Building Footprint	(71, 71, 71)	⬛
Road	(172, 167, 167)	⬜
Pavement	(219, 219, 219)	⬜
Waterfront	(17, 176, 254)	🟦
Miscellaneous Land Use	(31, 142, 41)	🟩

Table 3.1: Summary of the categories, their corresponding colors (RGB), and actual color representation.

Figure 3.1 shows a sample from the generated dataset. Data points were sampled from the 25 largest cities in Sweden. The dataset was additionally supplemented with data points from three capital cities in Europe: Berlin, Paris, and London. Images containing less than 5% of pixels corresponding to buildings were removed from the dataset.

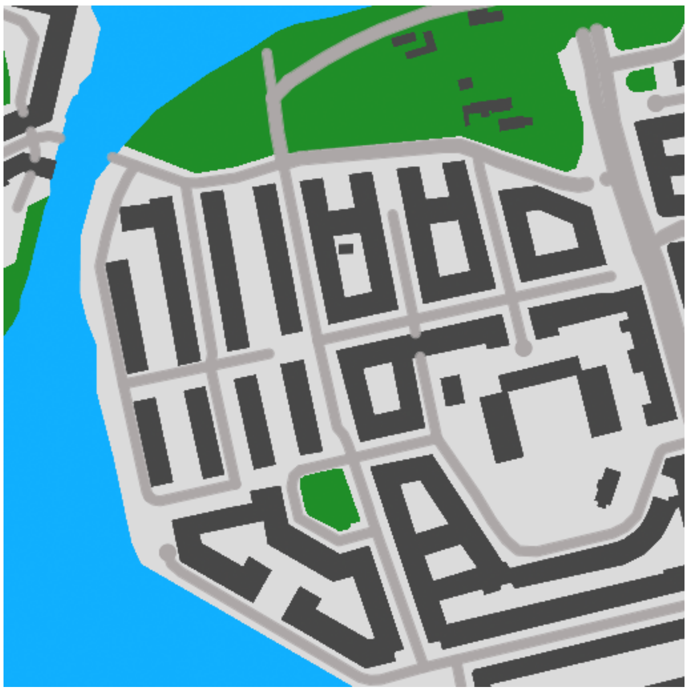


Figure 3.1: Datapoint corresponding to approximately 600x600 meters of urban area in Stockholm, Sweden. Obtained through OpenStreetMaps via Mapbox static image API.

The final dataset consists of almost 17000 images with 38% from cities in Sweden, 37% from London, 19.5% from Berlin, and 5.5% from Paris. The dataset was split into a training set of 12644 images, a validation and a test set of 2048 images respectively, and a small visual test set of 100 images.

3.2 Masking

Images were masked with a custom masking policy pertaining to the intended use case of the inpainting system. The masks resembled an undeveloped zone surrounded by an urban area. Masks were generated during training according to the principles outlined in section 2.2.2. The masking algorithm used a naive approach, referred to as Square Mask, of generating random rectangles of varying size to create input image-mask pairs. Generated masks have a sidelength of between 1/4 to 1/2 of image sidelength. This equated to masks that cover 6.25% to 25% of image pixels. In addition, pixels that correspond to waterfront were left unmasked based on the assumption that areas corresponding to water would not be developed, and were therefore not part of the intended use case.

A more sophisticated approach, referred to as Block Mask, was assessed on a limited scale as well, with the intention of generating more realistic inpainting scenarios. This algorithm extracted the road network of a given image and identified city blocks, then chose one of the blocks as the mask. In the context of the algorithm, a city block was defined as an area of non-road pixels surrounded by road pixels at all

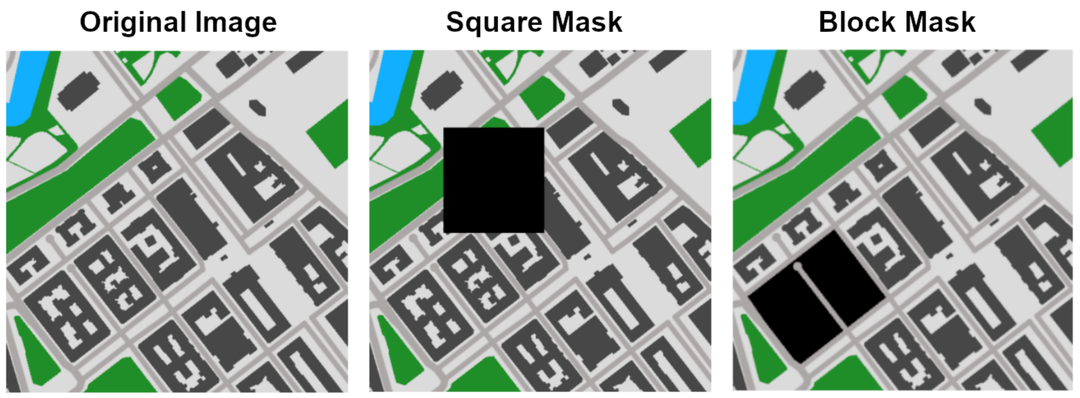


Figure 3.2: Left: Unmasked image from the gathered dataset. Center: Example of mask obtained with square mask algorithm. Right: Example of mask obtained with block mask algorithm.

sides (except the image boundary). Generated masks should cover between 2-25% and have a density ratio of at least 10%. Density is defined as

$$\text{density} = \frac{n}{M} \tag{3.1}$$

where n is the number of masked pixels corresponding to building and N is the total number of masked pixels. The algorithm generated masks iteratively until these conditions are met, otherwise defaulting back to the square mask algorithm. Figure 3.2 illustrates resulting mask generation from the two algorithms.

3.3 Transfer Learning

Transfer learning was conducted with a naive approach and with fine-tuning on the two models, as outlined in section 2.3, 2.4 and 2.5. The naive approach involved applying the models as is on the new data and evaluating the results to form a baseline. Both models used are available with pre-trained weights on GitHub and are written in Python with the Pytorch library. As Deepfillv2 was originally written with Tensorflow, a Pytorch reimplementation[30] was used. This was in order to maintain the consistency of interactions between code in the original repositories and external scripts. Models pre-trained on the Places dataset[31] were chosen as the prevalence of man-made structures, such as houses and rooms, would hopefully form a good basis for inference on the new dataset.

It was theorized that in order to successfully transfer knowledge from the new dataset on the pre-trained models without negatively impacting existing knowledge, the training scenario during fine tuning should resemble the training conducted in the original works. As such, data pre-processing such as normalization, cropping, and transformations were kept as is in the original training pipelines. In addition, hyper-parameters including learning rates were kept as is, and no layers were frozen during fine-tuning. Contrary to common practices where learning rates are often reduced and layers frozen during fine-tuning, this decision was made in the hope of avoiding disrupting the training stability of the GANs obtained by the original authors. The added benefit of this approach was that a time consuming hyperparameter search could be avoided.

During fine tuning Both LaMa and Deepfillv2 were trained on 512x512 pixel images cropped to 256x256 pixels. Models using block masking were trained on 512x512 pixel images resized to 256x256 pixels as cropped images contained to few blocks to present a meaningful training scenario. Only Deepfillv2 was trained using block masks due to time constraints.

During all training, due to GAN training being prone to instability, adversial and reconstruction loss was monitored and weights were saved periodically. After sufficient reconstruction loss was achieved, different states of the trained model were evaluated visually and, in some cases, by FID and LPIPS scores on validation data.

3.4 User Guided Density

The fine-tuned Deepfillv2 model was modified to allow the user to control the model's creative decision making. In particular, the user guided model was trained to incorporate instructions considering the desired *Floor area ratio* (FAR) of the generated area. As this thesis does not consider the number of storeys in a given structure, FAR would be defined as

$$\text{FAR} = \frac{m}{M} \tag{3.2}$$

where m is the number of pixels in the generated area corresponding to building and M is the total number of pixels in the generated area. In the context of this thesis, FAR is essentially calculated in the same manner as density, see equation 3.1. As a result, FAR will henceforth be referred to as input density and generated density in this thesis.

The original architecture of Deepfillv2 was modified to take input density as an additional input during training. During training, the input density was calculated as the ground truth or "real" density of the masked area. The density value was concatenated in the channel dimension to the output of the final layer before upsampling in the Deepfillv2 course generator. The aim was to provide additional information while allowing early layers to solely focus on the structural information contained in unmasked pixels. Figure 3.3 showcases this modification. Remaining layers were initialized with the fine tuned models weights and the entire network was additionally trained to allow the modified layer to relearn. By explicitly providing ground truth density as an input to the model and training it on data with varying density values, the model learned to associate different input density values with corresponding generated density values. To further incentivize the model to utilize this additional input, an MSE term was computed between the input density and the generated density and added to the total loss function as a regularization term. *Density MSE* is formulated as:

$$\text{Density MSE} = \frac{1}{n}\sum_{i=1}^{n}(D_r - D_g)^2 \tag{3.3}$$

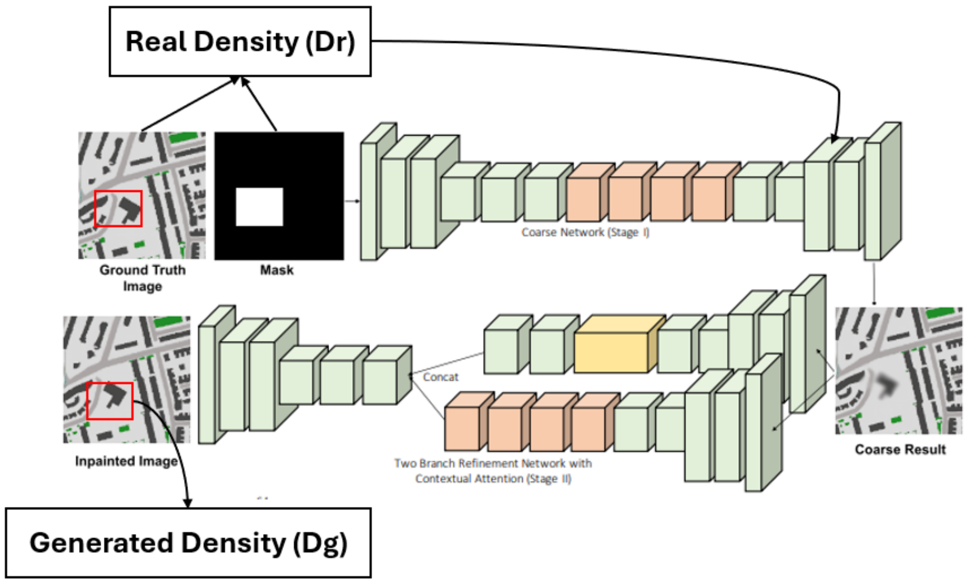


Figure 3.3: Illustration of modifications to the original Deepfillv2 architecture to enable user guided density. Image modified from [4].

where n is the number of data points , D_r is the input density and D_g is the generated density. Two user-guided models were trained: one incorporating Density MSE and one without density based regularization.

3.5 Evaluation

The models were evaluated visually and by calculating FID and LPIPS-scores on the test data. To avoid bias introduced by the feature extractors, care was taken when comparing FID and LPIPS scores. Since the dataset used is entirely new and has not been featured in any prior work, the FID and LPIPS scores only served as a comparison between the models obtained in this thesis. To keep consistency of evaluation, all FID and LPIPS scores were calculated on the entire, uncropped, or otherwise changed, test data with consistent masks. The user guided models were additionally evaluated by their capability of generating proposals with the desired FAR.

4 Results

This chapter will present and describe the results obtained including calculated FID and LPIPS metrics as described in section 2.6.1, 2.6.2 and 3.5, and visual examples of inpainted images generated by the different models described in chapter 3. All metrics and visual examples were generated on test data with pre-generated square masks for consistency of evaluation. In addition, a short section describing results obtained with Deepfillv2 fine-tuned using block masks is provided to provide a basis for discussion in Chapter 5.

4.1 Quality analysis

Results obtained from 6 different models are presented. The models compared are as follows:

- **Deepfillv2 Baseline:** The Deepfillv2 model applied directly on the test data.
- **Deepfillv2 Fine-Tuned:** The Deepfillv2 model fine-tuned using training data described in section 3.1.
- **LaMa Baseline:** The LaMa model applied directly on the test data.
- **LaMa Fine-Tuned:** The LaMa model fine-tuned using training data.
- **User Guided without MSE** The modified Deepfillv2 model trained **without** an additional density based MSE term, as described in section 3.4.
- **User Guided with MSE** The modified Deepfillv2 model trained **with** an additional density based MSE term, as described in section 3.4.

The obtained FID scores are presented in Figure 4.1. It is apparent that fine-tuning the models results in better FID scores with a significant performance increase between the baseline and fine-tuned models. This is especially noticeable when comparing baseline Deepfillv2 with fine-tuned Deepfillv2. The user-guided models suffer decreased performance compared to the fine-tuned Deepfillv2 model they are based on, with the model trained without additional regularization performing slightly better. The best performing model is clearly fine-tuned LaMa.

Figure 4.2 show the mean of LPIPS scores obtained for the models. When looking carefully, the same trends shown in Figure 4.1 are present however much more subtlety. With the exception of Deepfillv2 Baseline, all models received almost

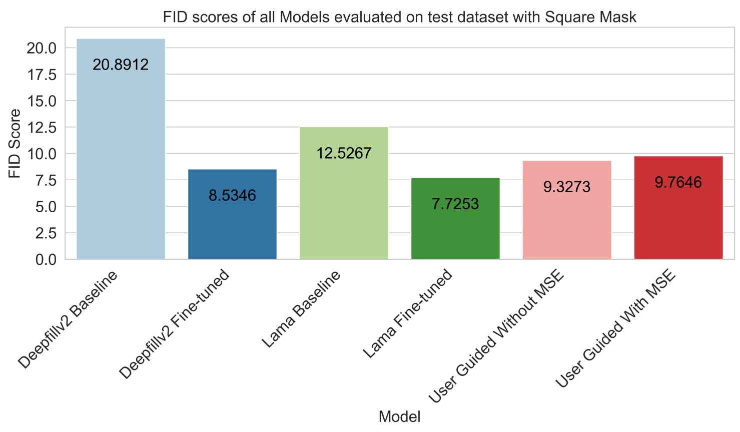


Figure 4.1: FID scores for all models trained and evaluated on square masked evaluation data. Lower scores indicate better quality.

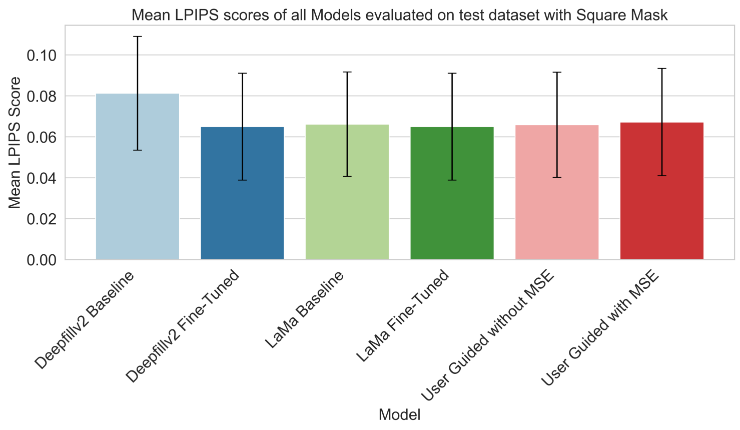


Figure 4.2: LPIPS scores for all models trained and evaluated on square masked evaluation data. Lowers scores indicate better quality.

identical LPIPS scores. The implications of this will be discussed further in chapter 5.

Visual examples of inpainted images are shown in Figure 4.3. These examples are only a small subset of the 100 images present in the visual test set and were chosen to showcase general trends in the inpainted images generated by the different models. For more visual examples, see Appendix 1.

Baseline Deepfillv2 struggles with capturing periodic structures present in the image and produces blurry and distorted inpaintings. The fine-tuned Deepfillv2 model, on the other hand, produces much sharper inpaintings with mostly clear distinctions between houses, roads, and ground while sometimes meshing houses and green (land use) pixels. The model struggles with generating straight lines and larger structures, often generating winding, organic looking, structures and small tetragons. The user guided models retain the characteristics of the fine-tuned Deepfillv2 model. Visually, the quality of the inpaintings are slightly less coherent compared to fine-tuned Deepfillv2 with inappropriately placed road segments being common, especially for higher values of input density.

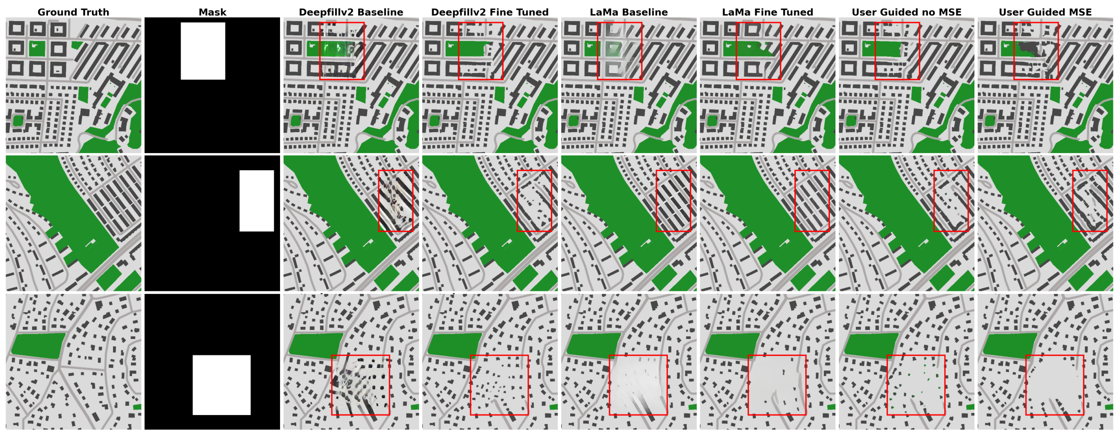


Figure 4.3: Visual samples for all models trained and evaluated on square masked evaluation data. Best viewed zoomed in.

The baseline LaMA model captures surrounding structures significantly better than DeepFillv2-based models. However, it struggles with overgeneralization, often producing blurry averages between different modes, especially when clear, periodic structures are absent in the image. The fine-tuned LaMa model is capable of producing sharp, clear structures reminiscent of houses in some cases, but is prone to produce low density inpaintings with few house structures when there are few surrounding periodic structures present to draw inspiration from. Still, fine-tuned LaMa shows the highest visual quality of all evaluated models.

4.2 Further Evaluation of User Guided Density Model

Visual Evaluation shows that in general, the user guided models produced inpainting of lower visual quality compared to fine-tuned Deepfillv2 when the value of input density differed greatly from the global density value of the input image. This is especially evident for high values of input density provided to the MSE regularized model, as shown in Figure 4.4.

Looking more closely at Figure 4.5, it is evident that the mean absolute error between input density and generated density remains quite consistent over the entire density range for the regularized model. Conversely, the unregulated model experiences increasing mean absolute error as input density increases. Visual evaluation confirms that this occurs as the model prioritizes generating context dependent content over adhering strictly to the desired density values.

Figure 4.5 shows the mean absolute error between input density and generated density of the user guided models generated over the entire test dataset for increasing input density values. The regularized model adheres more closely to the desired density value, as shown in Figure 4.5, but does so at the expense of generative quality, as seen in Figure 4.4.

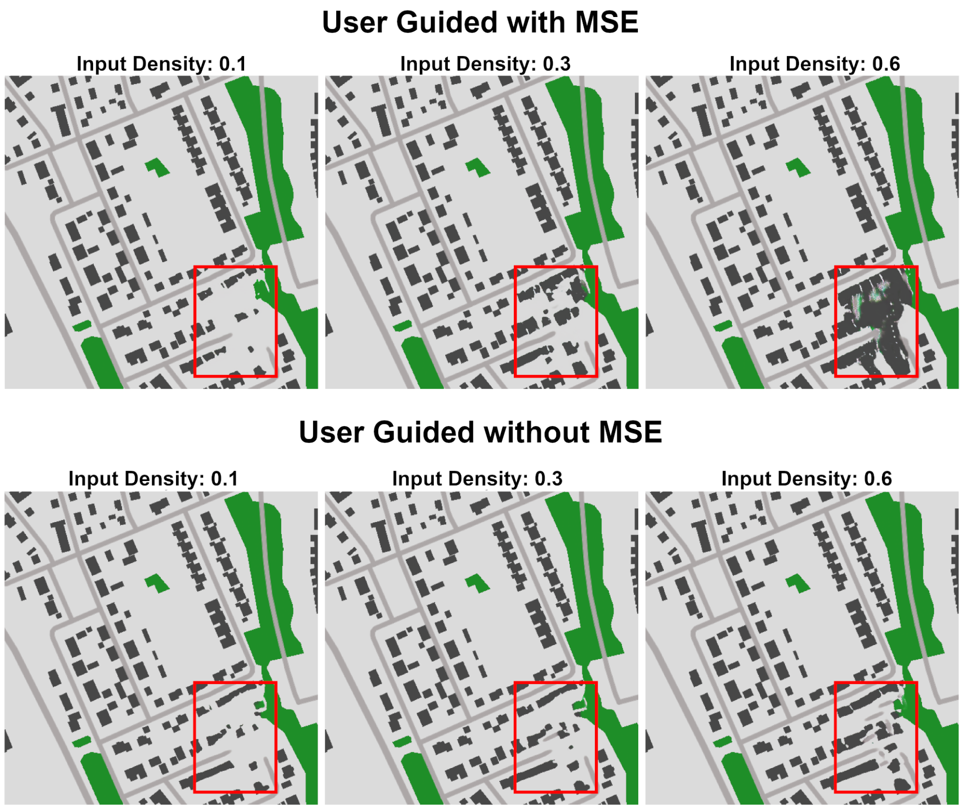


Figure 4.4: Top Row: Footprint suggestions generated with density regularized user guided model for increasing input density values. Bottom Row: Footprint suggestions generated with user guided model without density regularization for increasing input density values.

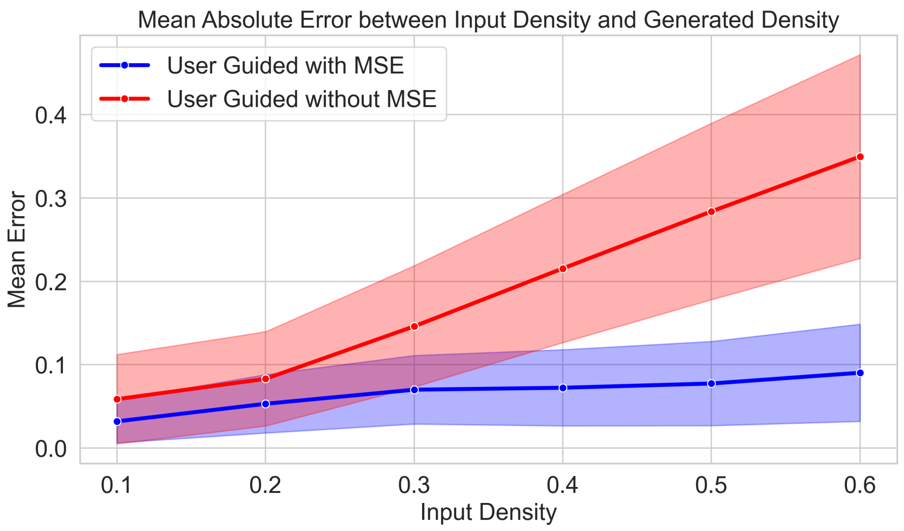


Figure 4.5: Mean absolute error between input density and generated density calculated over the entire evaluation data set for increasing input density values.

4.3 Block Mask Model

The fine-tuned Deepfillv2 model trained on block masks, described in section 3.2, was not able to generate any meaningful content. Training stability was not achieved and resulted in a mode collapsed model. Resulting inference on test data results in inpaintings consisting of random noise with no resemblance to the training data. This has important implications for the role that masking plays in creating a successful model, and will be discussed further in Chapter 5.

5 Discussion

This chapter will provide a discussion regarding the results presented in Chapter 4. Furthermore, it will provide some suggestions on how the methods could be improved in the future and discuss the suitability of fine-tuned inpainting networks to generate context dependent footprint suggestions.

5.1 Baseline and Fine-Tuned Models

It is evident, both by the visual inspection and analytical evaluation presented in section 4.1, that inpainting models can gain increased generative capabilities on previously unseen data by way of fine-tuning. It is somewhat unsurprising that the fine-tuned LaMa model achieved the most impressive results, as it is a larger and more capable model compared to Deepfillv2. Visual evaluation shows that although both models are capable of generating context dependent footprint suggestions, the suggestions are not of sufficient quality to fully satisfy the needs of the intended use case. As mentioned earlier, fine-tuned Deepfillv2 footprints are unrealistic and irregularly shaped, while fine-tuned LaMa footprints are more realistic but often sparse. It appears that the Deepfillv2 architecture is not capable of capturing the surrounding structure well enough to inpaint realistic looking footprints. The LaMa architecture on the other hand does have this capability, and it is possible that performance on the test data is not limited by network architecture but rather by training data and training regimen. Better performance could possibly be achieved by training multiple different models on subsets of the dataset. This would result in different models specialized on high density areas, low density areas, and so on. It is also possible that a proper hyperparameter search could improve results as well.

Masking approach likely plays a large role in the sparsity of fine-tuned LaMa predictions as well. The square mask algorithm generates masks in arbitrary positions. The resulting mask often partially covers footprints located at the border between masked and unmasked areas. During training, the model learns to rely on these partially covered footprints since continuing the structure into the masked area is a good way to achieve a low reconstruction loss. When the model is presented with a scenario that contains few or no structures to continue upon combined with no clear structural patterns to follow, the resulting suggestions are sparse or even empty. This also highlights an issue with using inpainting algorithms. A good strategy to complete an image is often to simply expand on the content present at the borders between masked and unmasked regions. This is especially evident when trying to apply a square mask trained model on block masked images. Since the masked region in this case is surrounded by roads at all sides, the model often inpaints the entire region with road texture.

Another strategy to improve performance could be to use a dataset with less complexity. Instead of using RGB images with different types of regions, one could use grayscale images that only show the building footprints. This would allow the network to focus entirely on generating building footprints without becoming confused by surrounding information. Although this could improve the model's capability of generating realistic footprints, ignoring structures such as roads would limit the model's applicability since such information could be important to consider in real life. Transfer learning could be less effective as well, as grayscale images differ more significantly from the RGB images the networks are originally trained on.

5.2 Block Masks

The Block Mask algorithm was intended to solve the issue of models relying on partially masked footprints and to train and evaluate the models on a scenario that more closely resembles a real life use case. Why training on block masks was unsuccessful is not entirely known but mask diversity could be a possible explanation. Square masks are diverse as there are few restrictions on how images are masked. Subsequently, there are essentially infinite amounts of image-mask pairs that could be generated. On the contrary, there are only so many city blocks present in any given image of the dataset. Data quantity might be to low as the networks are exposed to the same image-mask pair repeatedly during fine-tuning. This might explain the training instability and mode collapse. A possible solution could be to introduce some random masking elements on top of the block masks. This could provide the intended benefits of using the block masks while solving the issue of mask diversity.

5.3 User Guided Models

It is apparent that using additional regularization is important to make the models adhere more closely to the desired density. More interesting is the fact that regularization is not entirely required. As the input density is calculated from the ground truth pixels of the masked area, it serves as valuable information to the network when trying to lower reconstruction loss. As a consequence, the networks learn to consider this information even when it is not strictly regularized to do so. FID scores suggest a limited decrease in quality when using additional regularization, implying some sort of trade-off between adhering to user input and footprint quality. This quality decrease is, subjectively, not apparent in visual evaluation.

The density based user guided models are a good proof of concept of how inpainting networks can be modified to consider additional user inputs. Any other metric that can be extracted from the image during training could potentially be similarly inserted into the network. This includes density values for other structure types such as roads or parks but possibly other information such as the number of desired buildings or storeys (if applicable) in the generated area. It could also be of interest to investigate how inserting input density into different parts of the network would affect the results.

5.4 Evaluation

FID proved to be a useful metric in evaluating the images, with clear distinctions between different models' performance that match the visual evaluations. Although this is not a particularly robust way to ensure that FID generates meaningful metrics, it would appear that the feature extractor is capable of extracting relevant features from the dataset used in this thesis.

LPIPS on the other hand does not generate relevant metrics, as the LPIPS scores in all models except the baseline Deepfillv2 were very similar. It would appear that the feature extractor used in LPIPS is not adept at extracting meaningful features from the new dataset. Another explanation is that the learned weights obtained by training on real and fake images labeled by humans do not assign relevant importance to the features extracted from different layers. If the LPIPS algorithm was to be trained on the new dataset, it could potentially provide more relevant evaluation metrics but this could be problematic as the dataset is quite small.

5.5 Method Suitability

Although the models investigated have shown promise in generating footprint suggestions, there are some drawbacks and limitations of the method used in this thesis in creating a suitable system for the intended use-case. The biggest drawback is the fact that the models used in this thesis are entirely deterministic. Given the same image-mask pair, the models will always output the same prediction. The value of a generative model for footprint suggestions in the early stages of urban planning lies in the ability of producing many different suggestions for the same area in short amounts of time. This would allow an architect to consider and be inspired by a substantial amount of suggestions before making the final decision. The inpainting networks used in this thesis are not capable of providing this. Experiments with introducing noise to the latent space could be conducted, but it is unclear how it would affect the inpainting capability of the model or the possibility of fine-tuning. Another solution would be to use other types of inpainting networks, such as those that utilize diffusion models [17].

As mentioned earlier, training on square masks does not result in a model capable of generating suggestions for a realistic masked area, therefore further experiments with block masks or other masking approaches would be needed. Finally, raster images are inherently unsuitable for use in urban planning applications, as most of the graphics used in this field are in fact vector images. This could prove to be a challenging issue to overcome as most research on computer vision tasks, image inpainting included, is conducted on raster images.

6 Conclusion

This thesis has shown that it is possible to create a generative model capable of generating context dependent footprint suggestions to some extent. Although it shows promise, the methods used in this thesis should be refined in order to create a generative model capable of creating more realistic footprints in increasingly complex scenarios. More recent and capable networks should form the basis of further research if possible and additional data could be gathered. The method could also likely be applied to other formats of data, such as formats where different information such as noise pollution and air quality are present in additional channels in order to create more informative suggestions.

Furthermore, this thesis shows that transfer learning is applicable on inpainting networks and result in increased performance on the task of generating context dependent footprint suggestions. Transfer learning proved to be a resource effective way of adapting these models, resulting in significantly shorter training times and smaller data requirements. Standard inpainting appears to be a task that forms a good basis for the task of generating footprint suggestions.

Finally, it has been shown that inpainting networks can be modified to accept user guided density inputs. This provides a good proof of concept for a method that could be adapted to other types of information, resulting in a model that gives the user increased control of the generative process.

Bibliography

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[2] Y. Wang, C. Wu, L. Herranz, J. van de Weijer, A. Gonzalez-Garcia, and B. Raducanu, "Transferring gans: generating images from limited data," 2018.

[3] M. B. Pont and P. Haupt, *Spacematrix: Space, Density and Urban Form.* TU Delft OPEN Publishing, May 2023.

[4] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-form image inpainting with gated convolution," 2019.

[5] W. Li, Z. Lin, K. Zhou, L. Qi, Y. Wang, and J. Jia, "Mat: Mask-aware transformer for large hole image inpainting," 2022.

[6] S. J. Quan, "Urban-gan: An artificial intelligence-aided computation system for plural urban design," *Environment and Planning B: Urban Analytics and City Science*, vol. 49, p. 239980832211005, 05 2022.

[7] S. Fedorova, "Gans for urban design," 2021.

[8] D. J. Eck, *Introduction to Computer Graphics.* Open Textbook Library, 2016.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016. `http://www.deeplearningbook.org`.

[10] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, "Image inpainting based on deep learning: A review," *Displays*, vol. 69, p. 102028, 2021.

[11] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," 2021.

[12] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," 2021.

[13] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[14] Y. Kossale, M. Airaj, and A. Darouichi, "Mode collapse in generative adversarial networks: An overview," in *2022 8th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, 2022.

[15] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," 2018.

[16] H. Zheng, Z. Lin, J. Lu, S. Cohen, E. Shechtman, C. Barnes, J. Zhang, N. Xu, S. Amirghodsi, and J. Luo, "Cm-gan: Image inpainting with cascaded modulation gan and object-aware training," 2022.

[17] Y. Wang, C. Cao, and K. F. X. X. Y. Fu, "Towards context-stable and visual-consistent image inpainting," 2024.

[18] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5505–5514, 2018.

[19] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Trans. Graph.*, vol. 36, July 2017.

[20] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *Computer Vision – ECCV 2016*, (Cham), pp. 702–716, Springer International Publishing, 2016.

[21] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018.

[22] L. Chi, B. Jiang, and Y. Mu, "Fast fourier convolution," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 4479–4488, Curran Associates, Inc., 2020.

[23] E. O. Brigham and R. E. Morrow, "The fast fourier transform," *IEEE Spectr.*, vol. 4, pp. 63–70, Dec. 1967.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[25] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.

[26] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[27] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *CoRR*, vol. abs/1801.03924, 2018.

[28] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ." `https://www.openstreetmap.org`, 2017.

[29] Mapbox, "API Reference | Mapbox Static Images | Mapbox." Available from: `https://docs.mapbox.com/api/maps/static-images/` (Accessed: 2024-9-20).

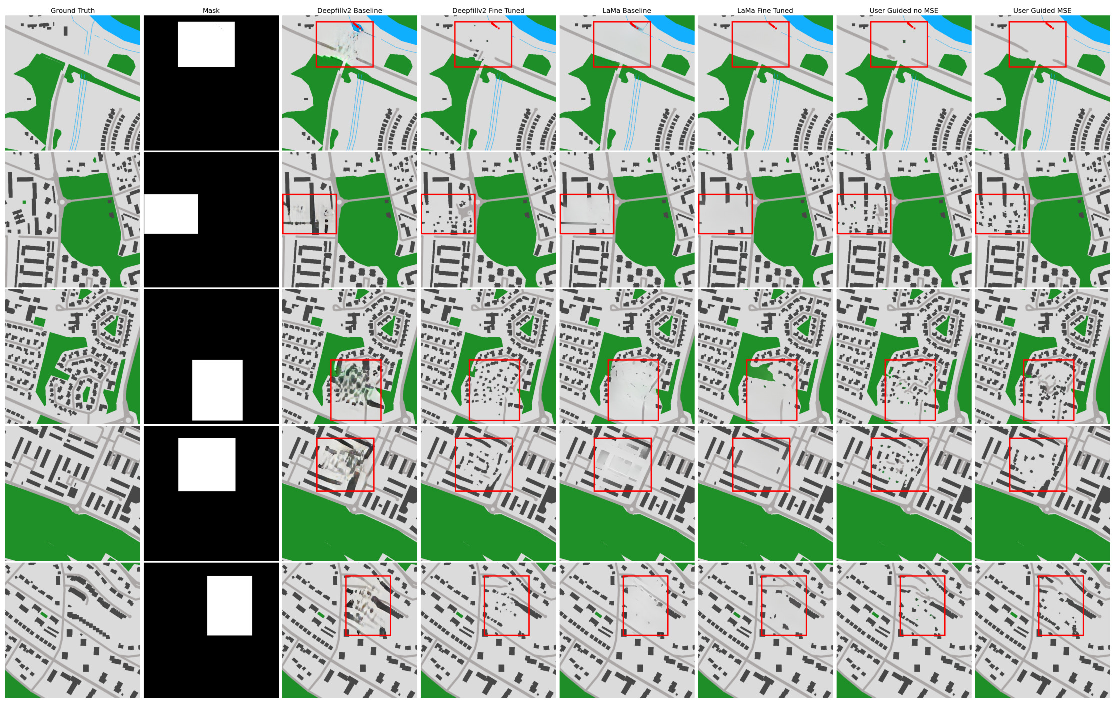
[30] L. Nippert, "deepfillv2-pytorch." `https://github.com/nipponjo/deepfillv2-pytorch`, 2024.

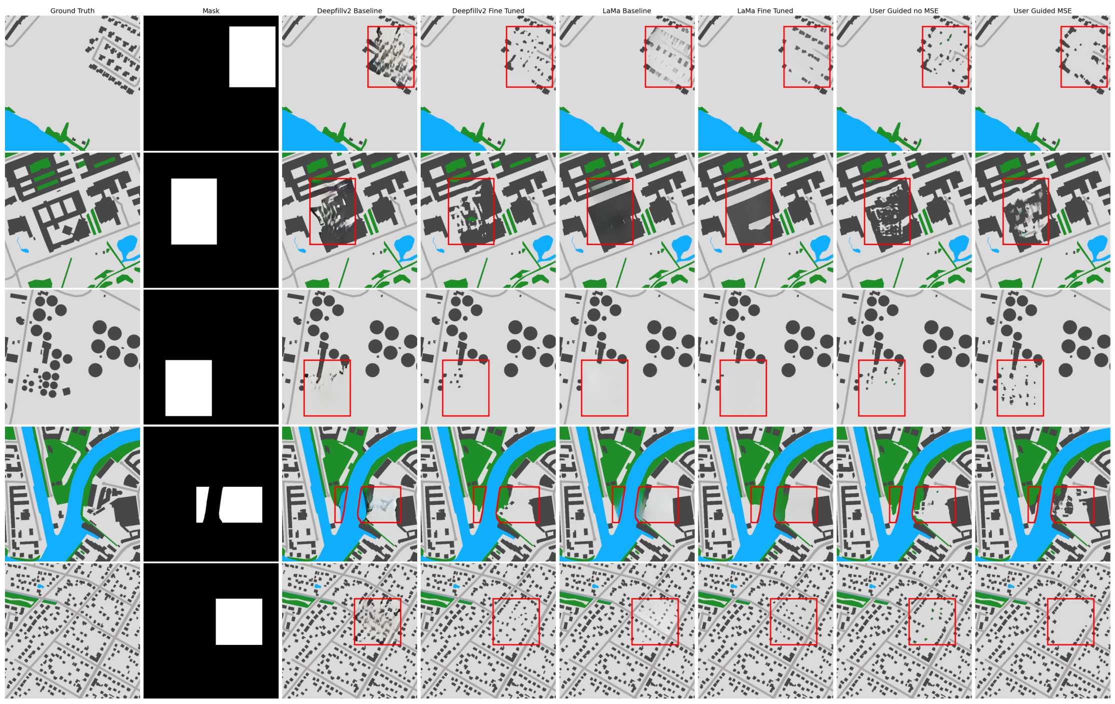
[31] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

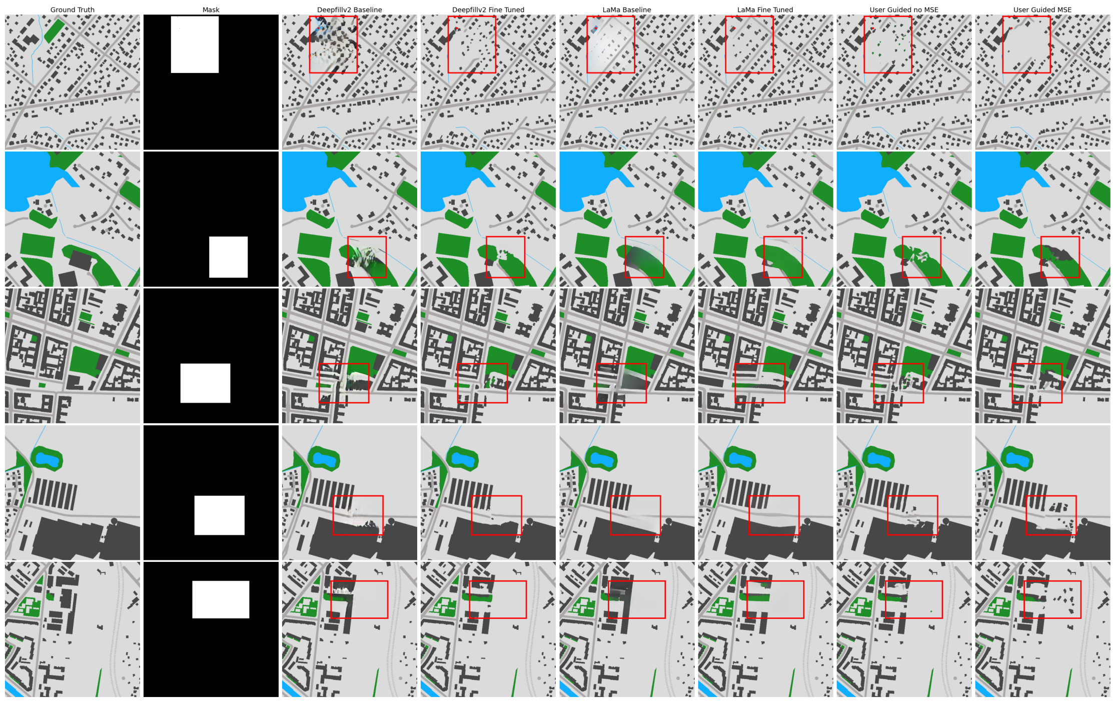
A

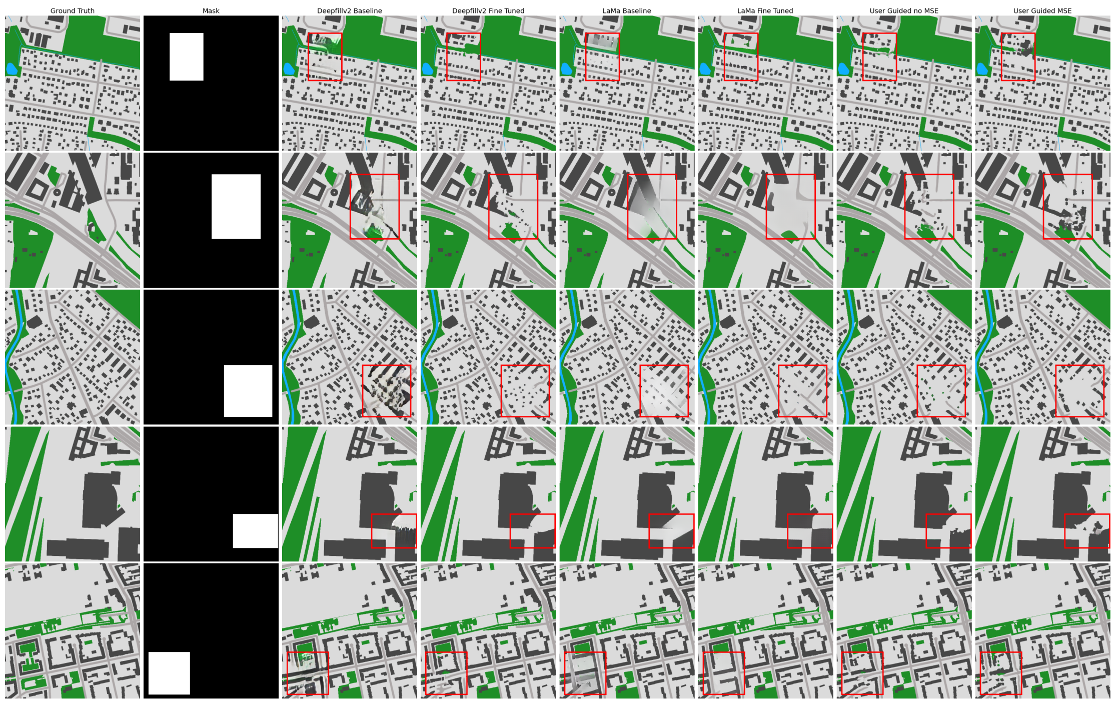
Appendix 1

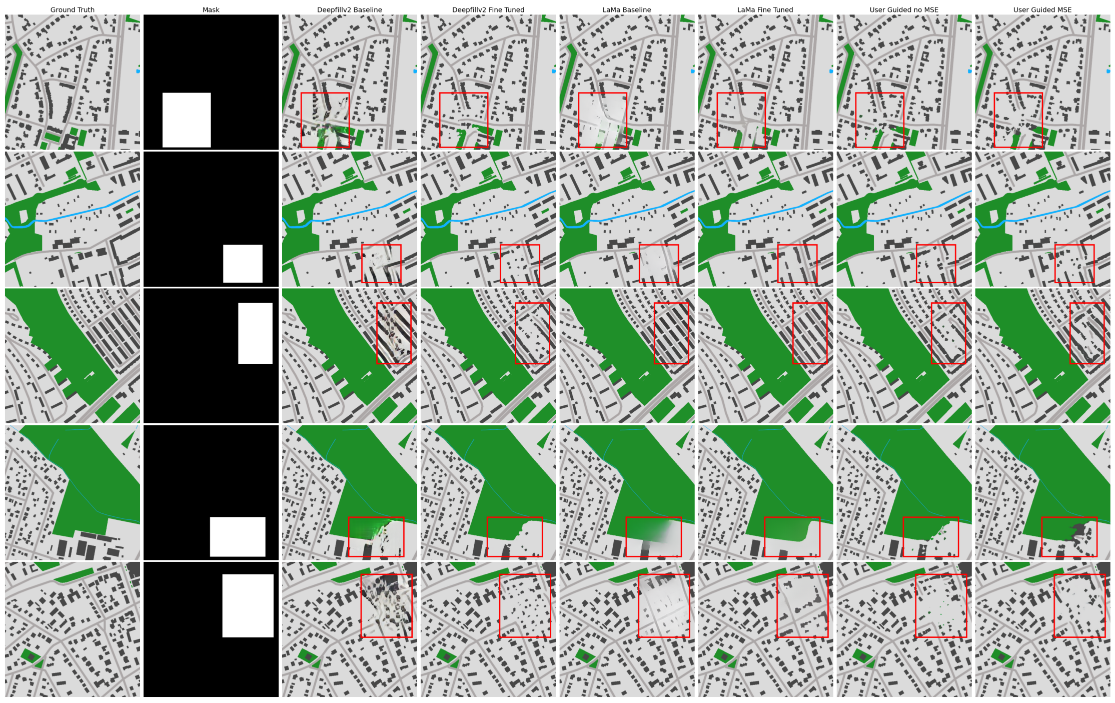
This appendix shows visual results on the entire visual test dataset consisting of 100 images for the 6 models described in section 4.1 Best viewed zoomed in.

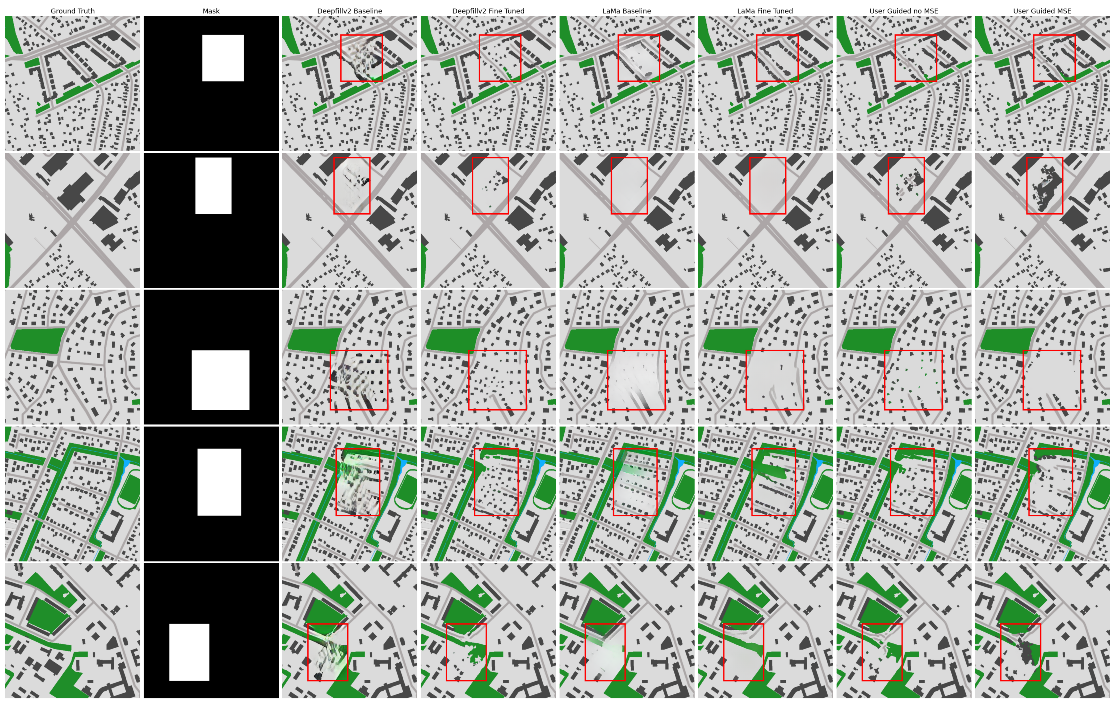


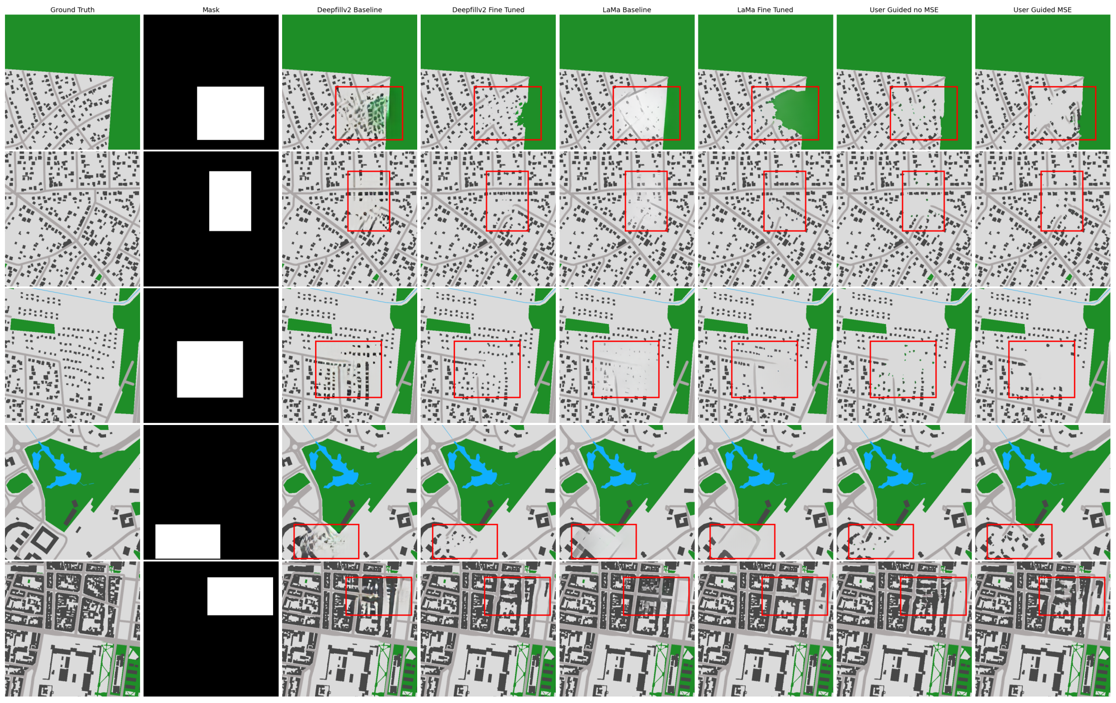

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

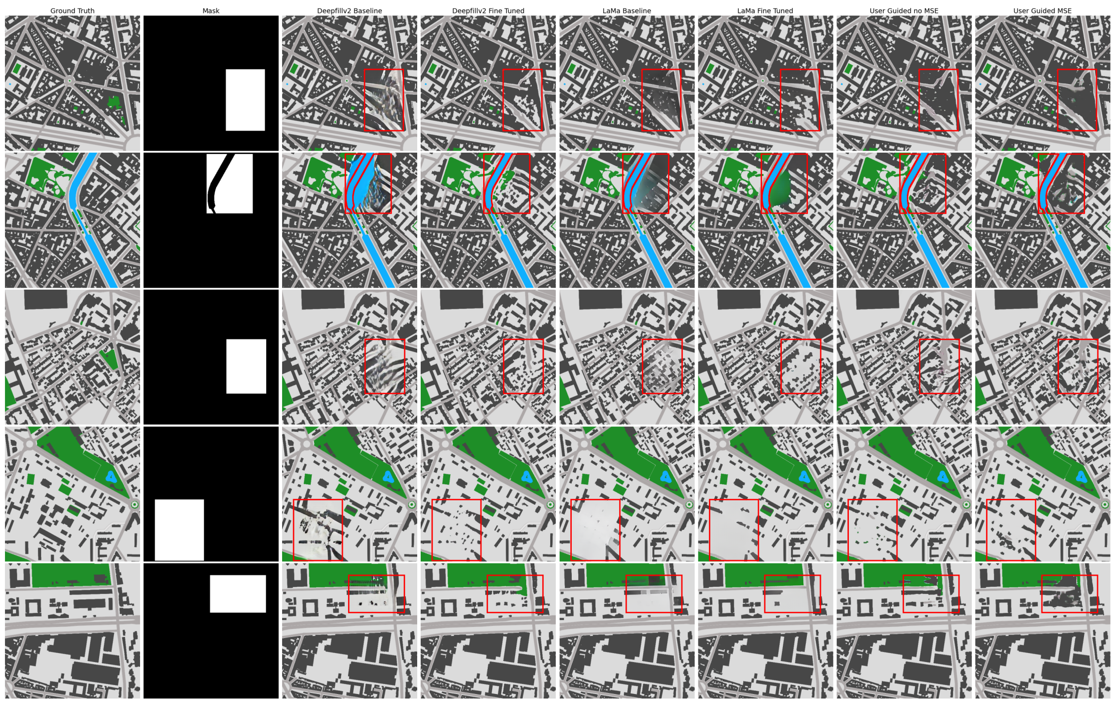

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

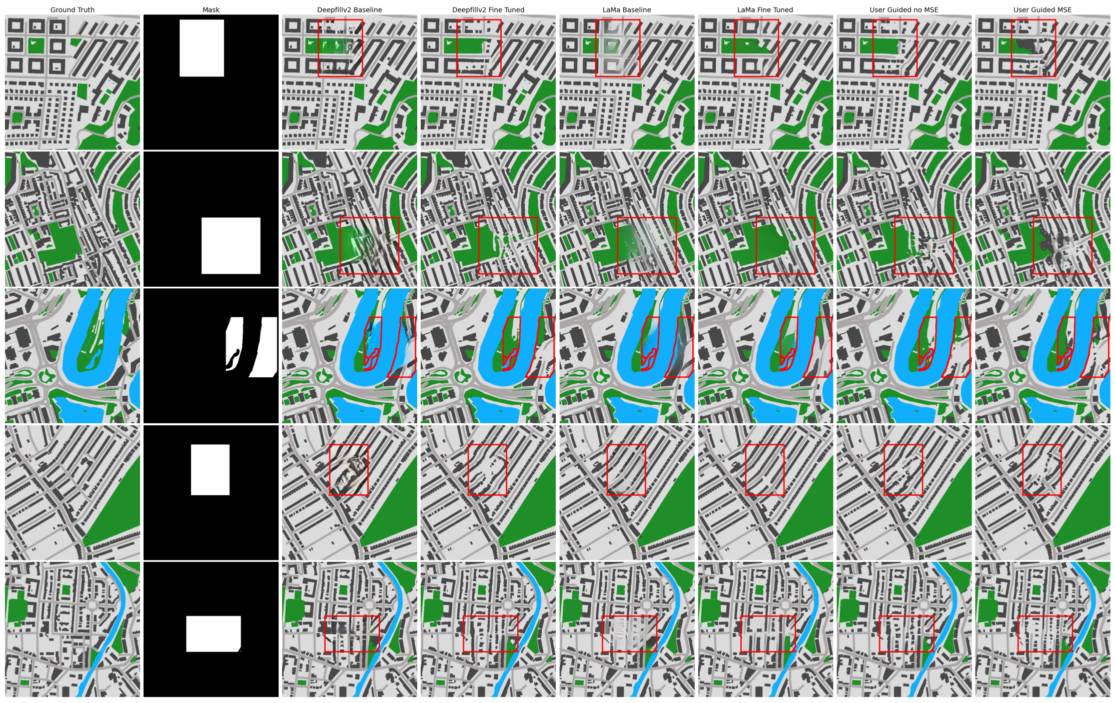

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

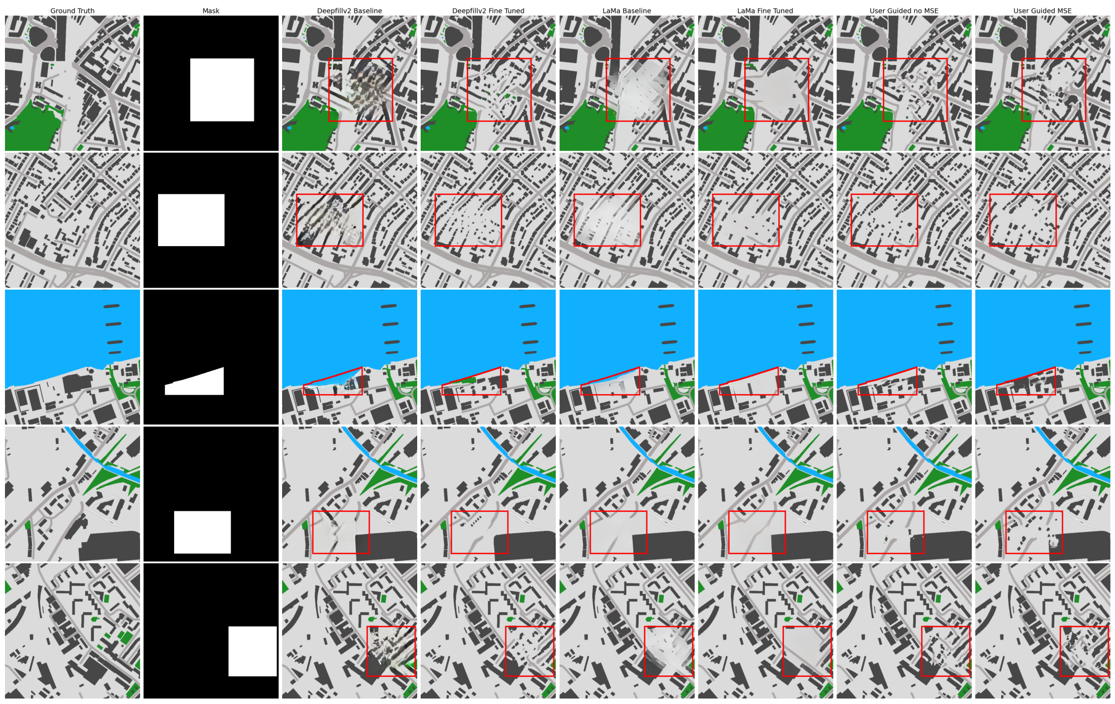

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

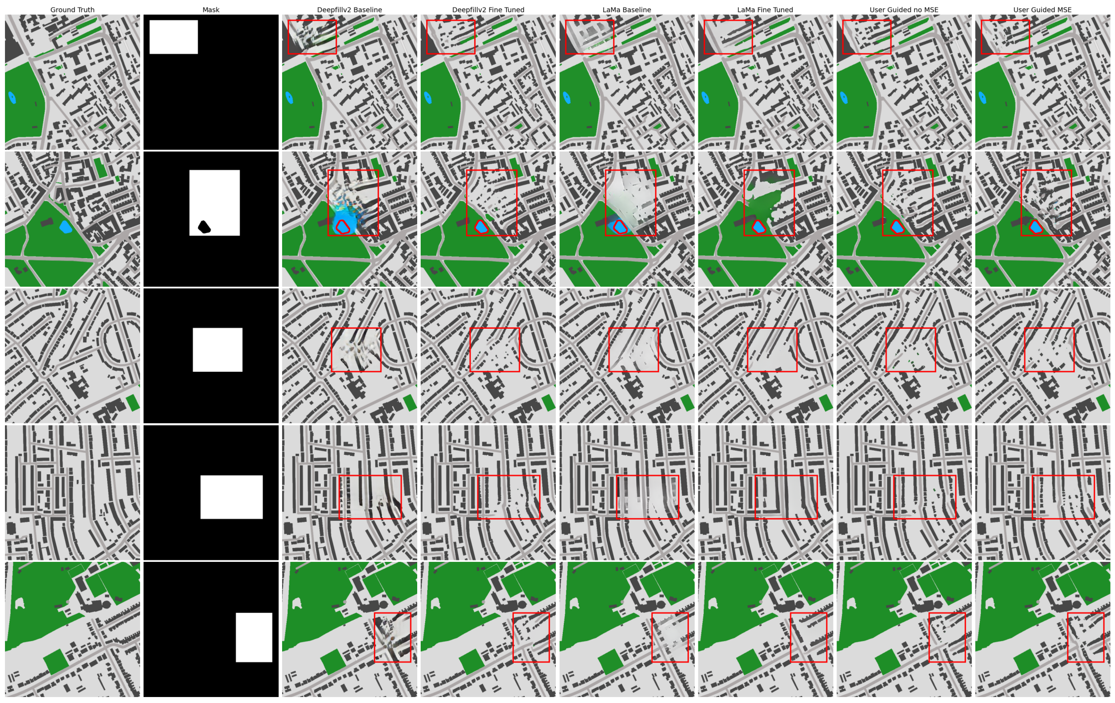

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

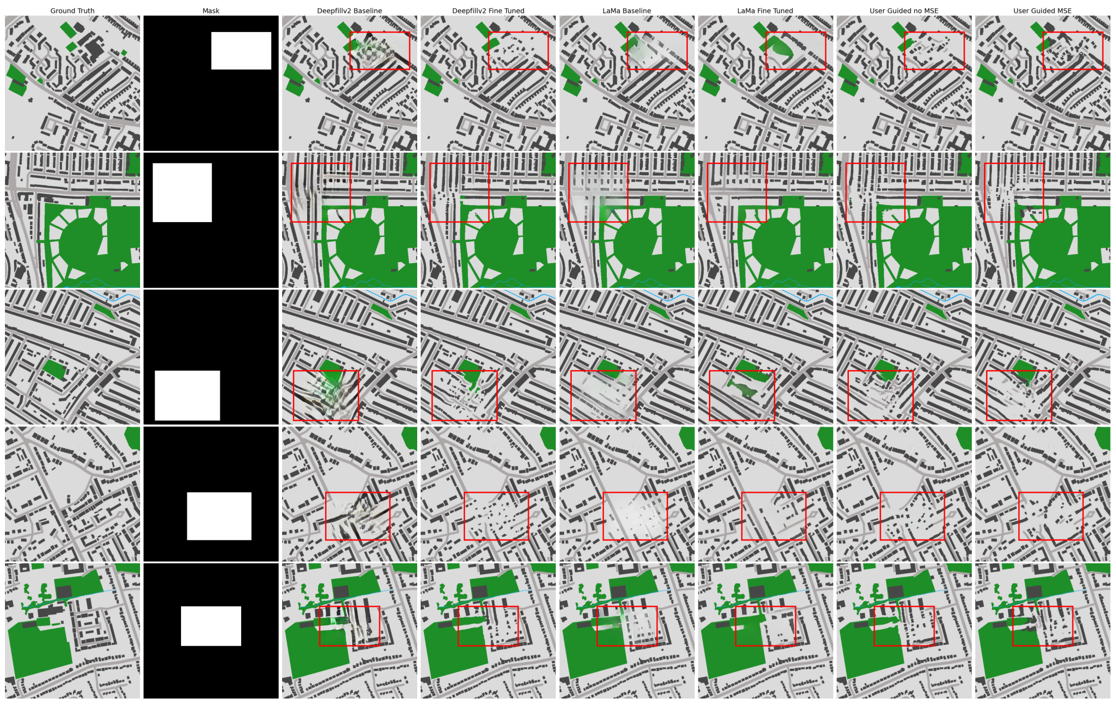

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

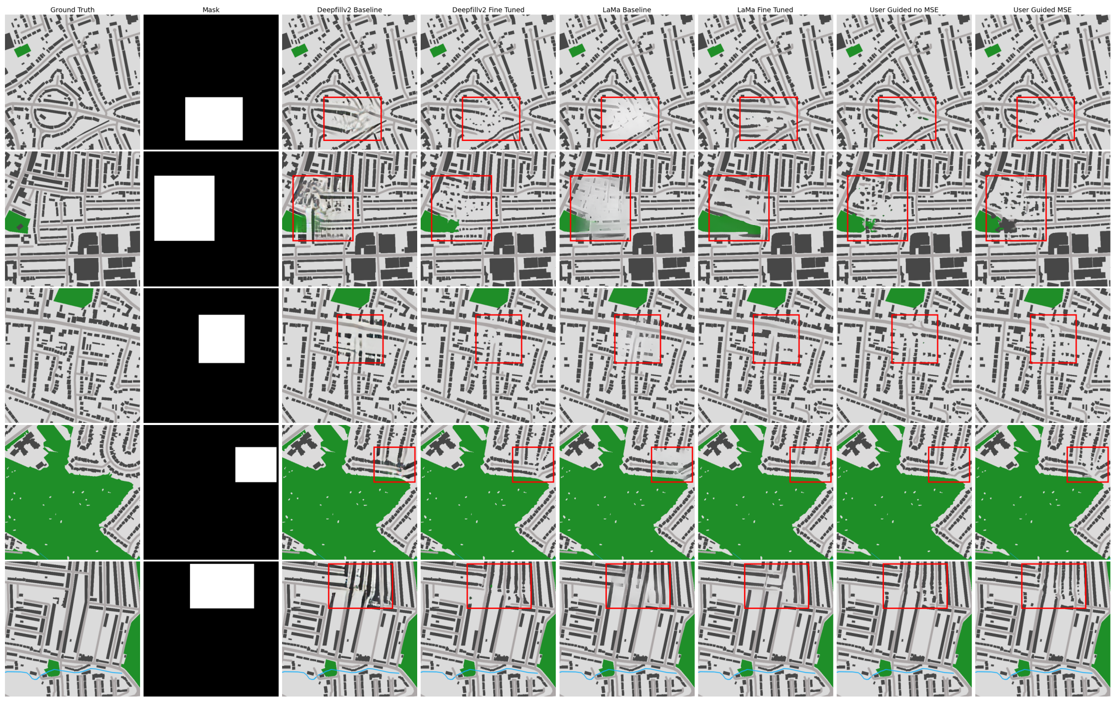

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

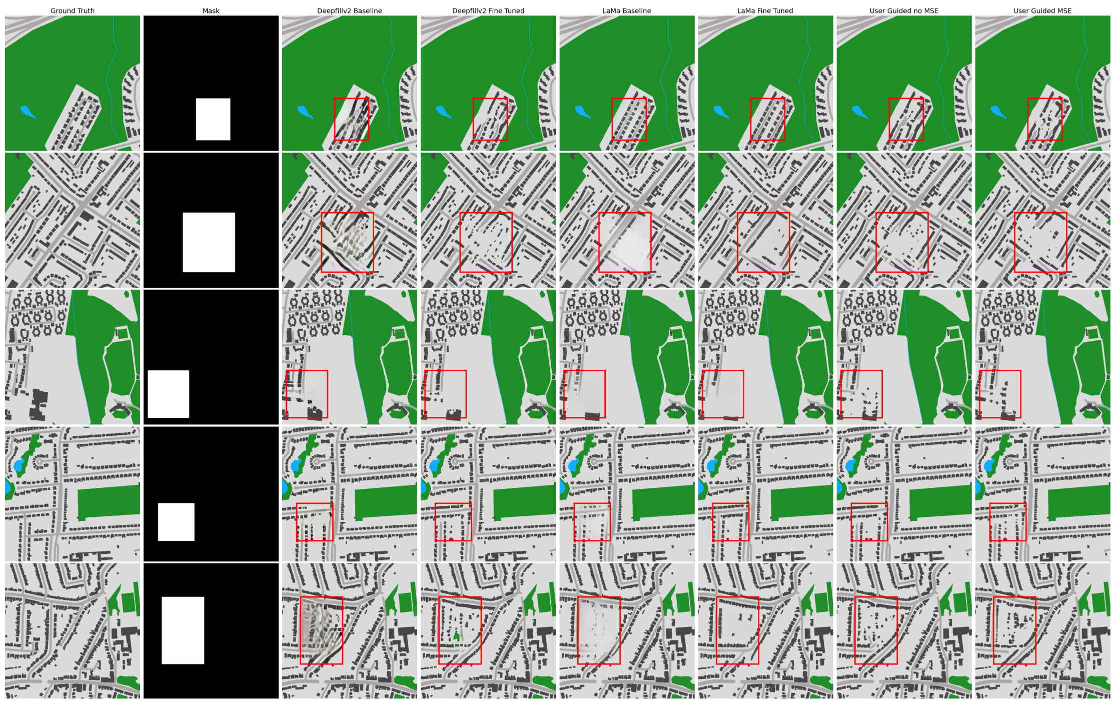

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

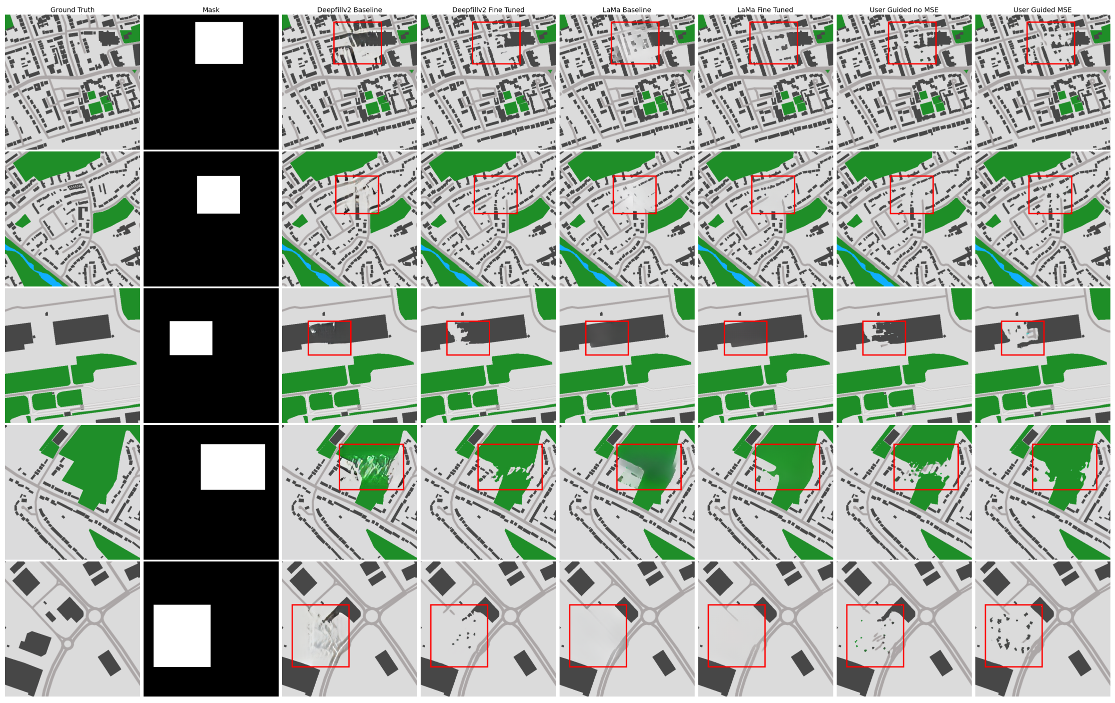

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

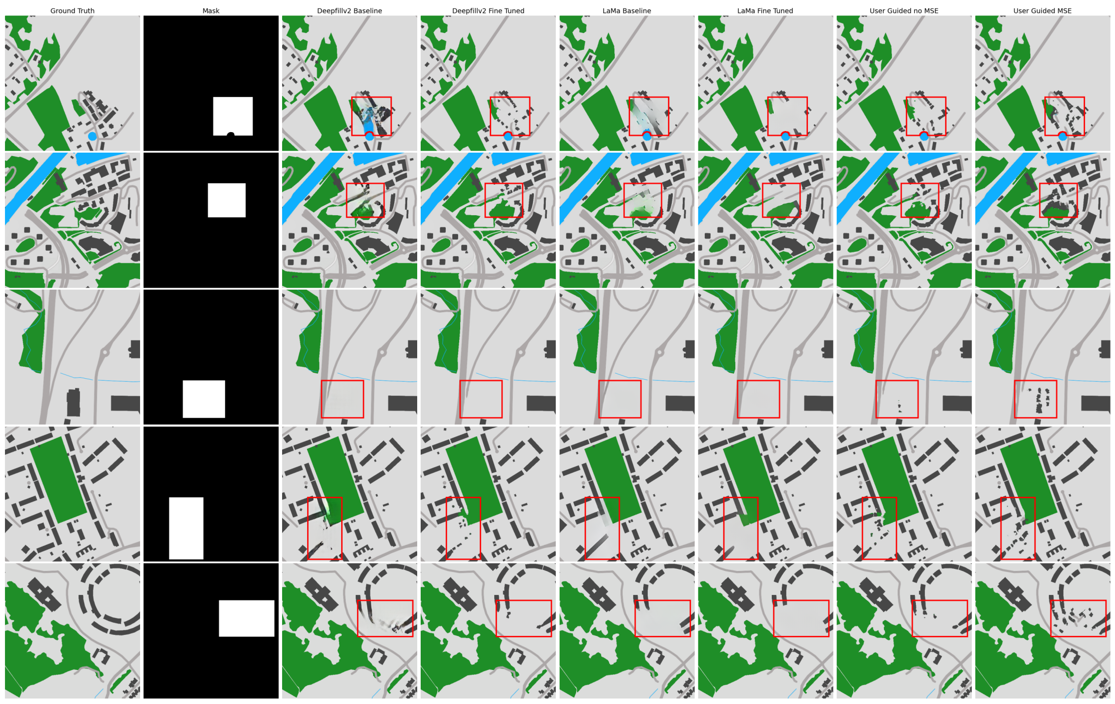

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

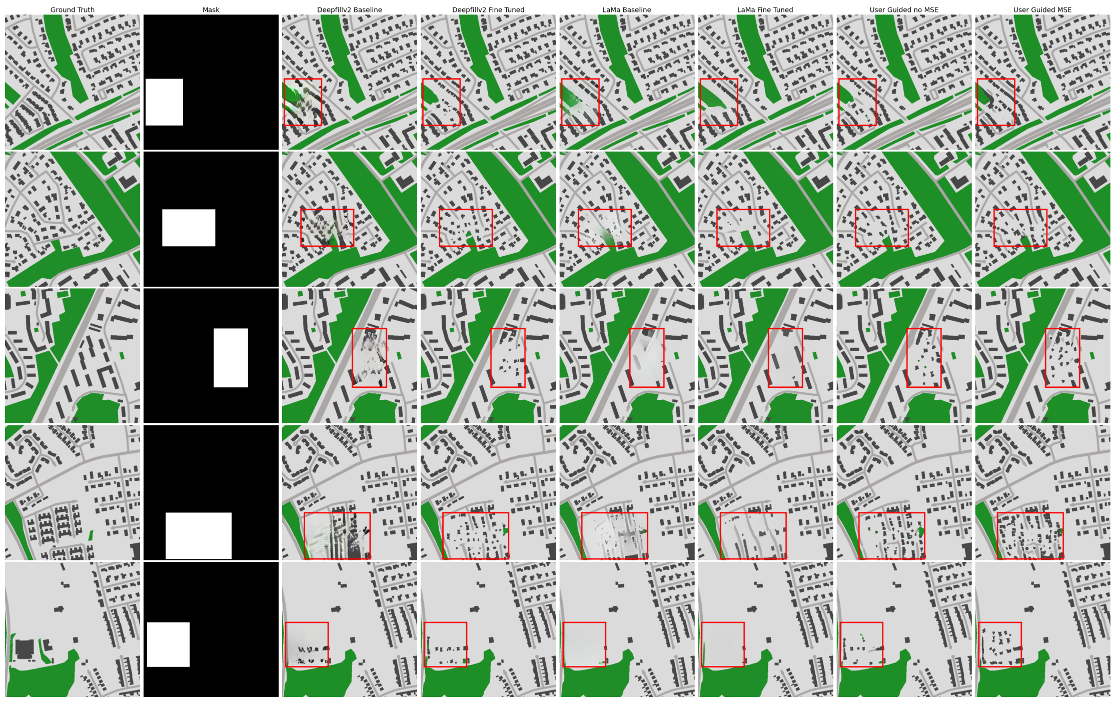

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

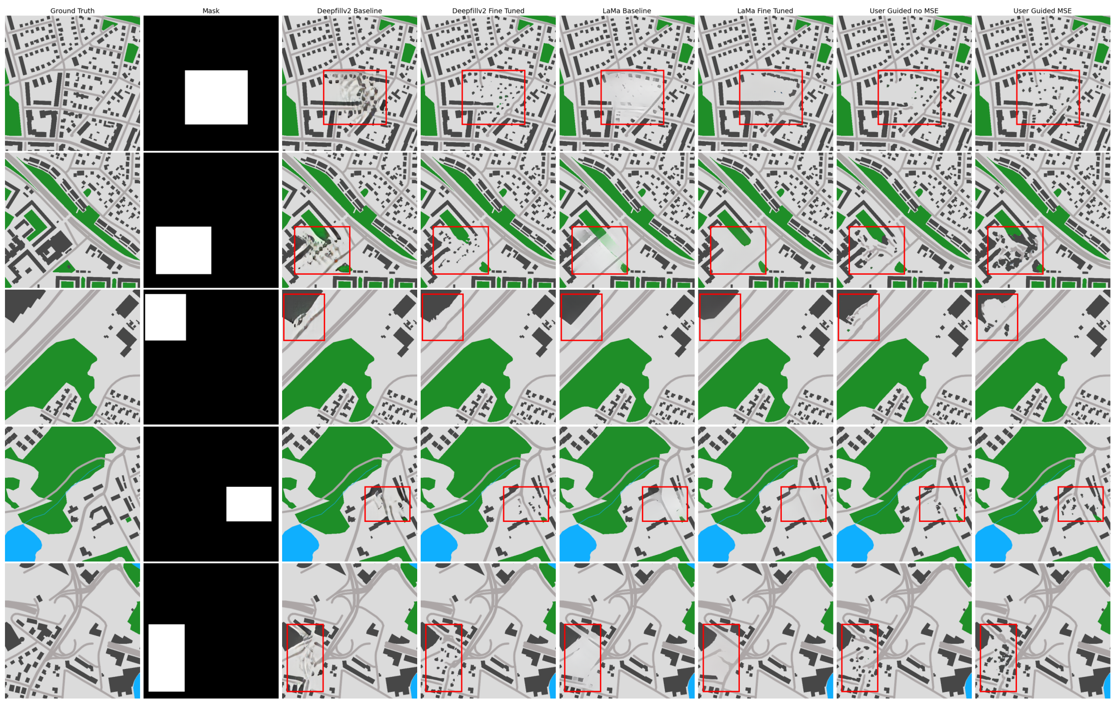

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

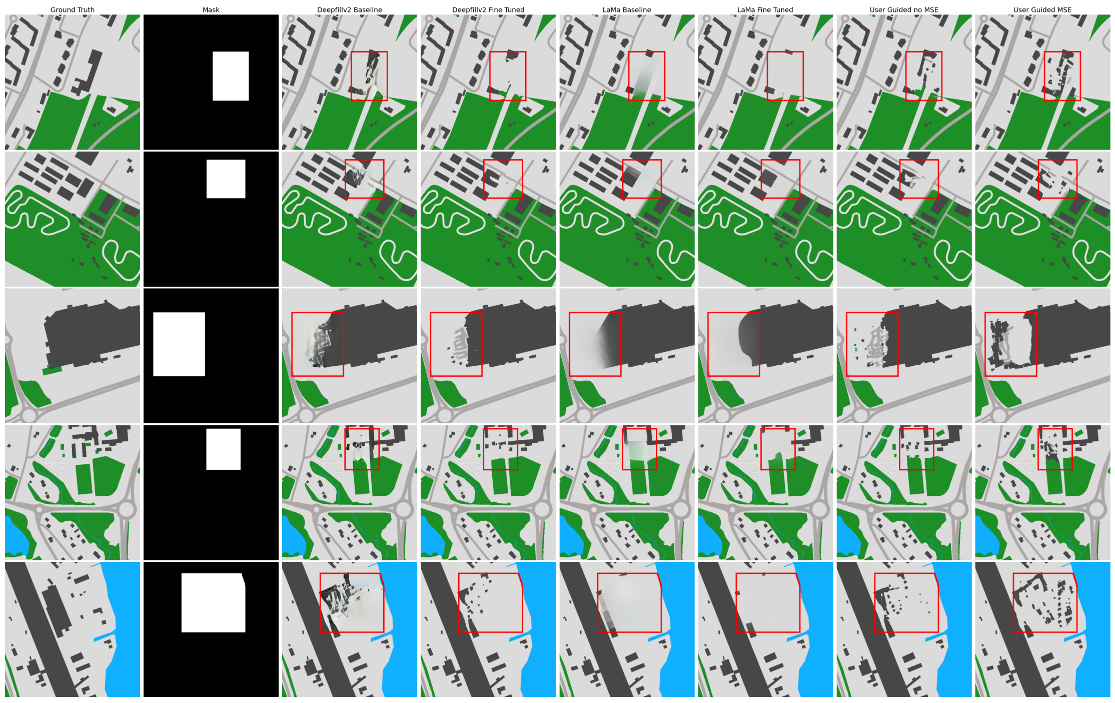

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE

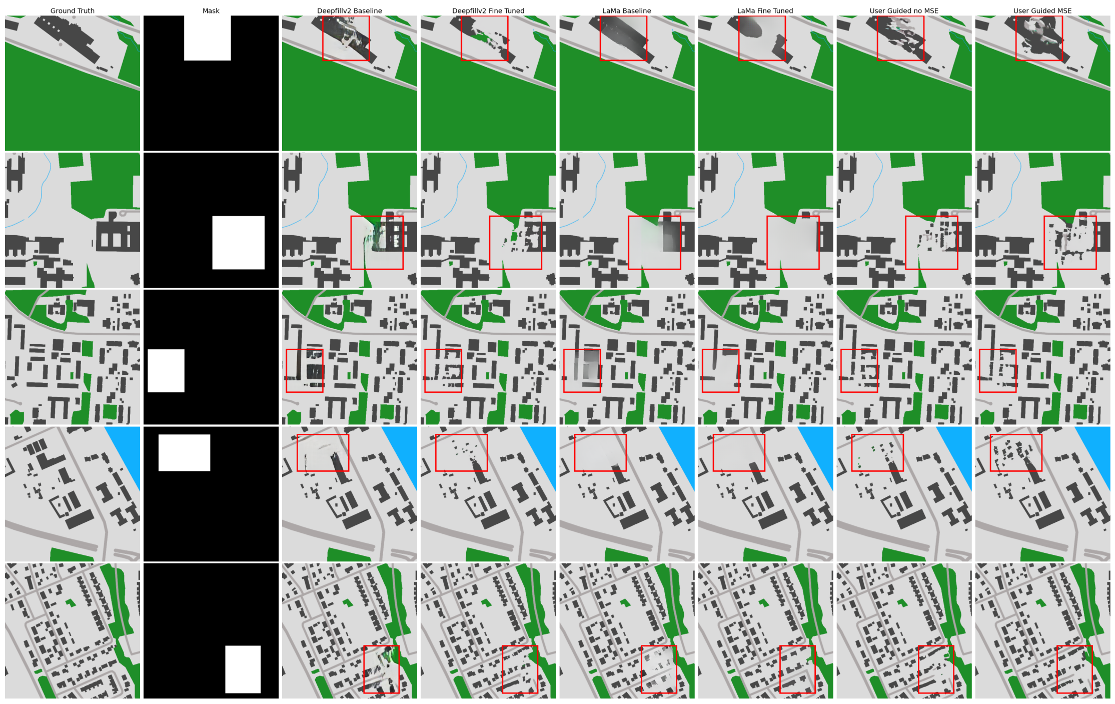

Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE


Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE


Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE


Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE


Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE


Ground Truth
Mask
Deepfillv2 Baseline
Deepfillv2 Fine Tuned
LaMa Baseline
LaMa Fine Tuned
User Guided no MSE
User Guided MSE