Image Inpainting Through Multi-Scale Partial Convolutions

Mainak Sarkar
Electrical and Computer Engineering
A59019511

Rounak Sen
Electrical and Computer Engineering
A59020344

Vancheeswaran Vaidyanathan
Electrical and Computer Engineering
A59011835

Abstract

Partial Convolution layers are integral to deep learning-based inpainting methods. However, existing approaches often use them straightforwardly, resulting in limitations such as blurry textures and a lack of global structural information in the inpainting results. To address this, we propose the Multi-Scale Feature Extraction and Fusion (MSFEF) module. It comprises parallel convolution layers with different kernel sizes, which are weighted and fused to combine global and local information around missing pixels. We integrate the MSFEF module into the encoder of a simple encoder-decoder architecture. Through experiments on Helen, Stanford Cars, and a subset of LSUN datasets, we demonstrate the superiority of our approach both quantitatively and qualitatively. Remarkable inpainting results are achieved even with a simple encoder-decoder architecture, showcasing the effectiveness of the MSFEF module. An executable Colab notebook for this project can be found at this **link**.

1 Introduction:

In the field of Computer Vision, image inpainting constitutes an important class of image-to-image translation problems. The objective of image inpainting is to recover the missing pixel values in corrupted and damaged images, in a way such that the regenerated image carries visual and contextual meaning. Ideally, the best inpainting results are attributed to the instances when the restored image looks realistic and plausible. This has numerous real-life applications such as the restoration of historical photos in museums, restoration of old videos, removal of unwanted objects from the images (such as logos and watermarks), and so on.

In recent times, the field of inpainting has witnessed significant advancements with the emergence of deep learning models, enabling remarkable breakthroughs in addressing complex and challenging inpainting problems. One key module that has become an integral building block of many recent deep learning-based inpainting methods is the **Partial Convolution Layer**, which was originally introduced by Liu et al. in [1]. The primary reason behind the widespread adoption of partial convolution layers is due to their ability to limit the standard convolution operation to valid pixels within an image. Moreover, the associated mask update mechanism, also proposed by the authors of Partial Convolution Layers in [1], allows for the progressive filling of missing pixels as each partial convolution layer is applied. By incorporating a sufficient number of these layers, it becomes possible to progressively fill in all the missing regions of an image.

Many recent deep learning-based inpainting methods have adopted a straightforward approach when it comes to incorporating partial convolution layers: namely they include only a single partial

convolution layer in each stage for processing the input feature map. However, this approach proves insufficient when dealing with images containing significant missing portions or sparsely structured elements, as mentioned in [1]. In this report, we reason that these limitations arise due to the straightforward method of implementation, which results in the lack of an appropriate combination of global and local information in the inpainting of the missing pixels. To address these deficiencies, we propose a novel solution: the **Multi-Scale Feature Extraction and Fusion** (**MSFEF**) module.

The MSFEF module comprises three parallel partial convolution layers with different kernel sizes. These layers capture semantic information at different scales (global and local) surrounding the missing pixels. The output feature maps from these layers are fused using a layer-wise weighted linear combination, where the weights are learned during training. The fused feature map is then multiplied with the updated binary mask of the partial convolution layer with the smallest kernel size to produce the final output feature map of the MSFEF module.

To showcase the efficacy of the MSFEF module, we integrate it into the encoder section of a simplified encoder-decoder architecture designed for image inpainting. Through extensive experimentation on publicly available datasets such as Helen [2], Stanford Cars [3], and a subset of LSUN dataset [4] (Church Outdoors images), we demonstrate significant improvements quantitatively and qualitatively. The incorporation of the MSFEF module yields impressive inpainting results even with a straightforward encoder-decoder architecture. During training, we employ pixel-wise MSE, Perceptual Loss [5], and Style Loss [5] (both based on VGG-16), and an ablation study confirms the necessity of all three losses for optimal training. Furthermore, substituting single Partial Convolution Layers for the MSFEF module in the encoder section produces inferior results, highlighting the dominance of our approach.

Finally, we acknowledge certain limitations in the inpainting methodology that we have employed. One limitation is the need to appropriately adjust the kernel sizes of the partial convolution layers in the MSFEF module based on the resolution of the input image. This adjustment ensures proper incorporation of local and global semantic information into the inpainting results. For our future work, we propose potential approaches to overcome these limitations.

The project presents significant contributions aimed at enhancing the utilization of partial convolution layers in deep learning-based inpainting methods. These are our **key contributions**, which can be summarized as follows:

- Firstly, we introduce the Multi-Scale Feature Extraction and Fusion (MSFEF) module, for addressing the drawbacks of the current deep-learning based inpainting methods which include partial convolution modules in a straightforward manner. These models fail to perform well for large missing areas and images with sparse structures.
- Through extensive experimentation that we perform on three publicly available datasets, we show that with the application of the proposed MSFEF model, when applied to a simple encoder-decoder architecture, achieves significant results. We proved this via our ablation study and the results shown in later sections.
- Lastly, we identify certain limitations in the employed inpainting methodology and propose strategies to overcome these limitations. These findings will undoubtedly serve as a foundation for our future research endeavors, and for fostering further advancements in the field.

2 Related Works:

The existing inpainting methods can be broadly classified into two categories: non-learning approaches, and the more widely used deep learning methods. A brief literature survey of each of these categories is as follows:

2.1 Non-Learning Based Approaches:

Inpainting methods that do not rely on learning include various techniques such as pixel interpolation, diffusion, and traditional patch-based methods. These approaches heavily rely on the image statistics of the available parts of the image to perform image inpainting. In this domain, Bertalmio et al. [6] suggested for the first time, the use of a diffusion-based inpainting method, while Chan and Shen

[7] proposed variational models based on *Total Variation* to inpaint the corrupted regions of the image. These diffusion-based methods failed to produce accurate results when the corrupted area was large. The PatchMatch algorithm, proposed by Barnes et al. [8] attempts to fill the missing regions of an image by finding patches with high similarity from the undamaged regions. However, these algorithms (along with other conventional approaches) often failed to produce contextually reasonable inpainting results due to their lack of deep understanding of the available regions of the image.

2.2 Deep Learning Based Methods:

Deep learning models have a distinct advantage over traditional methods in their ability to extract high-level semantic information from an image, making them potentially much more effective in an inpainting task. Therefore, with the rapid development of deep learning techniques, and the introduction of Generative Adversarial Networks (GANs) [9], deep learning models have been applied extensively in image inpainting. Pathak et al. in [10], proposed a CNN-based context encoder model for image inpainting. Although the model showed good potential, it was not adept at generating fine textures and was only capable of fixing holes in the 64*64 central rectangular region of the image (it resized all the input images to 128×128, if the dimension was not 128×128 already). Yu et al. [11] formulated a model similar to [12], however, they replaced their post-processing step with a refinement network containing contextual attention layers. Although good results can be obtained, this method is quite computationally costly, and it fails to solve the semantic blur problem of images. Yan et al.[13] introduced a model called Shift-Net, where they incorporated shift connection layers in the U-Net architecture [14], in order for the model to obtain the global structural information of the image. Recently, Wan et al. [15] applied a transformer for image inpainting for the first time, but at the cost of increased computational complexity. Coming to GANs, most image inpainting methods lack control during image synthesis, resulting in deterministic transformations. To address this issue, Zheng et al. [16] and Zhao et al. [17] proposed a VAE-based network that trades off between diversity and reconstruction. Zhao et al. [18] also introduced a modulated convolution layer for the inpainting task, inspired by the StyleGAN2 [19]. Recently, a new family of autoregressive methods [20, 21, 22] has emerged that can handle irregular masks, providing a powerful alternative for free-form image inpainting. In a different direction, Richardson et al. [23] successfully inpainted missing regions using the StyleGAN [24] prior. However, similar to super-resolution methods [25, 26], this approach is limited to specific scenarios like faces. Ulyanov et al. [27] demonstrated that the structure of an untrained generator network contains an inherent prior that can be used for inpainting and other applications.

3 Proposed Method:

Several recently proposed deep-learning based inpainting models such as [28], [29], [30], and [31], employ partial convolution layers. The integration of these layers is characterized by a straightforward approach, where the intermediate feature map (or input feature map) undergoes a single partial convolution layer during each convolution stage in the model architecture. However, this simplified implementation method entails certain drawbacks, as extensively discussed in the original partial convolution paper authored by Liu et al. (2018). The primarily observed disadvantages can be primarily categorized into two types.

- In the case of images with a significant number of missing pixels, network architectures that incorporate partial convolution in a straightforward manner often fail to generate satisfactory inpainting results. The outcomes tend to exhibit blurriness and lack meaningful interpretation.
- One of the limitations of current inpainting methods is their difficulty in accurately restoring intricate details within images that have sparse structures. These details may include delicate elements like window grills or bars on a gate. Despite the incorporation of partial convolution techniques, these methods often struggle to effectively restore such thin and intricate structures, leading to suboptimal results in these specific cases.

Our hypothesis suggests that the limitations mentioned above can be attributed to the straightforward approach used in the implementation of partial convolution. This approach involves generating

the output feature map through the partial convolution operation utilizing a single kernel size. As a result, the integration of global and local information surrounding each missing pixel is not effectively achieved. When a large kernel size is utilized at a specific partial convolution stage, only global information is captured, while local information such as textures, edges, and contours is not appropriately extracted. Conversely, using a small kernel size leads to the neglect of global semantic information, including structural details and spatial coherence around the missing pixel values.

To deal with these disadvantages, we propose a simple Multi-Scale Feature Extraction and Fusion (MSFEF) module, through which we shall be able to incorporate both global and local information in the estimation of the missing pixel values, and thereby obtain inpainting results of superior quality. So, in this section, we shall describe the following topics one by one: First, we shall provide a mathematical description of partial convolution layers. Secondly, we shall discuss the high-level architecture of our simple encoder-decoder model which we shall implement, and the block-level architecture, operation and intuition behind our proposed MSFEF module. Finally, we shall provide the mathematical formulation of the loss functions which shall be used for training our deep network.

3.1 Mathematical Description of Partial Convolution:

As discussed before, in each partial convolution layer, the masked convolution operation is performed first using the input feature map and input binary mask, followed by the mask update step.

Let **W** be the convolutional filter weights of a particular output channel of the convolution output, and let **b** be the corresponding bias (scalar value for a particular output channel). Now, let $\mathbf{X}_p$ be the vectorized feature map values for the current position of the convolution kernel's sliding window, where p denotes the pixel position of the input image on which the current sliding window is centered (i.e. the results of the convolutional operation will be assigned to to the position p). Finally, let $\mathbf{M}_p$, be the vectorized mask values (having the same shape as $\mathbf{X}_p$) corresponding to the sliding window position centered at the same position p in the input binary mask.

The partial convolution operation at every position p is finally defined as follows:

$$x_p^{'} = \begin{cases} \mathbf{W}^T(\mathbf{X}_p \circ \mathbf{M}_p)\frac{sum(\mathbf{1})}{sum(\mathbf{M}_p)} + \mathbf{b}, & \text{if } sum(\mathbf{M}_p) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

, where $x_p^{'}$ denotes the convolution filter response assigned to position p of the input feature map. Additionally, $\circ$ denotes the element-wise multiplication operation, and $\mathbf{1}$ denotes a tensor with the same shape as M_p but with all the elements set to 1. The purpose of $\frac{sum(\mathbf{1})}{sum(\mathbf{M}_p)}$ term is to serve as a scaling factor to normalize the convolution filter response at point p, in order to adjust for the varying amount of available pixels (in sliding windows centered at different positions of the input feature map).

Following this convolution step, mask updation of the input binary mask is performed. The mask update step is given as follows:

$$m_p^{'} = \begin{cases} 1, & \text{if } sum(\mathbf{M}_p) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

, where $m_p^{'}$ denotes the updated mask values at position p of the input binary mask.

The convolution operation and mask updation step utilized in the partial convolution layer, as discussed above, remain identical to the approach presented in [1].

3.2 Network Architecture and the MSFEF Module

3.2.1 Network Architecture:

Our network's architecture is shown by 2 figures. Let us first discuss the high-level/overall architecture of the encoder-decoder model that we have implemented.

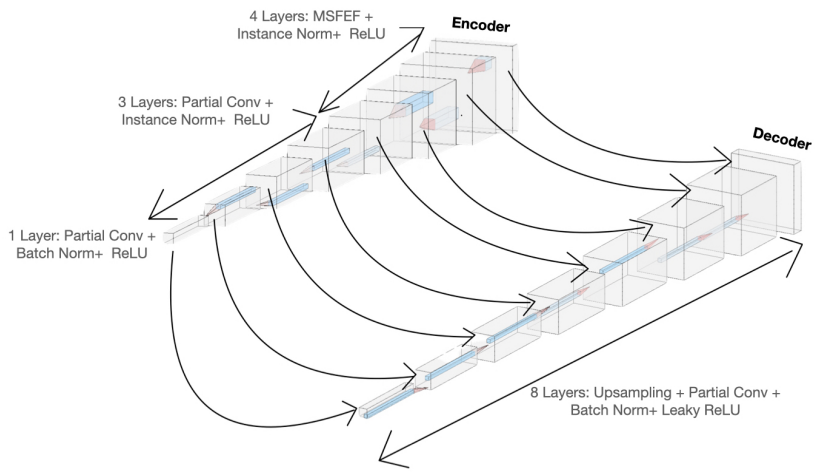

Figure 1: High-level Network Architecture: The architecture consists of two main parts, the Encoder and Decoder. The skip connections between them are represented by black curved lines. The Decoder block contains 3 subsections, the upsampling section, concatenation and partial convolution.

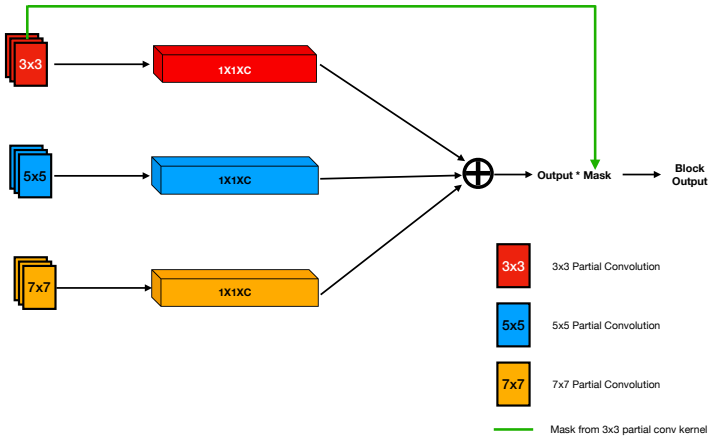

Figure 2: Low-level Network Architecture: Here we have 3 partial convolution kernels of sizes 3x3, 5x5 and 7x7. The mask is taken only from 3x3 partial convolution kernel and is indicated by the green arrow going from 3x3 partial convolution to the output section.

Figure 1 illustrates the high-level architecture of the image. The encoder section includes 8 convolution stages: the first 4 stages use the MSFEF module, and the remaining 4 stages use standard partial convolution layers. Downsampling by a factor of 2 occurs at each convolution stage (via strided convolutions).

Now, in this project, we use images (or resize them) of resolution 256×256 for all our experiments and as input to our encoder-decoder model. Due to downsampling at each convolution stage of the encoder, after the 4th convolution stage of the encoder, the feature map resolution becomes 16×16. Therefore, to capture global semantic information around the missing pixels, larger kernel sizes are unnecessary, as the receptive field has already been expanded due to the MSFEF module in the previous stages. Thus for reducing computation burden, only partial convolution layers are used from the 5th stage onwards. In contrast to typical encoder-decoder models, Instance Normalization is employed instead of Batch Normalization (see Figure 1) after each encoder convolution stage (except the last stage). This choice avoids the impact of placeholder values (e.g., 0) in missing pixels on the statistics used for normalizing valid pixels. Furthermore, Instance Normalization is beneficial for capturing fine-grained details, local characteristics, and improving contrast invariance in the encoder.

The decoder section comprises of 8 partial convolution layers and upsampling blocks, as depicted in Figure 1. The upsampling blocks use the Nearest Neighbor technique to upscale the height and width of the input feature map and mask by a factor of 2. Following the first upsampling block, the upsampled feature map and mask are concatenated with the output feature map and mask of the second-last convolution block in the encoder (respectively) along the channel dimension, via skip connections. The aggregated feature map and mask are then fed as input to the first partial convolution layer in the decoder. This process is repeated until the end of the decoder. The concatenation steps in all the stages of the decoder are denoted by black curved in later stages of the decoder are denoted by black curved lines in Figure 1, which represent the corresponding skip connections from the encoder to the decoder layer. After the upsampling layer in the final stage of the decoder, the upsampled feature map and mask are concatenated with the feature map and mask of the original network input and passed to the final partial convolution layer, whose output serves as the final output of our model.

The primary reason we do not use MSFEF layers in the decoder section is that in the encoder section, the MSFEF module is already used to appropriately encode the local and global information around the missing pixels, and therefore, the resultant feature map of the last stage of the encoder will already contain the local and global semantic information around each missing pixel. It is the job of the decoder to simply encode information in the final feature map of the encoder, and for this, it was our belief that partial convolution layers are sufficient.

For more clarity on the architectural design and specifications of our encoder-decoder model, please refer to Table 2 in the Appendix.

3.2.2 The MSFEF module:

Let us now delve into the block-level architecture of the MSFEF (Multi-Scale Feature Extraction and Fusion) module, which we have proposed and implemented in our encoder-decoder inpainting network. The architectural details are illustrated in Figure 2.

In this architecture, we simultaneously input the feature map of the MSFEF module to three parallel convolution layers with kernel sizes of 3×3, 5×5, and 7×7, respectively (as depicted in Figure 1). We chose these kernel sizes based on our initial intuition that they would enable the different partial convolution layers to capture semantic information at varying scales (from local to global) around each missing pixel location. We verified the validity of our intuition through experimental results.

(Note that the selection of these specific kernel sizes is suitable for 256×256 images. For higher-resolution images, larger kernel sizes may be required to effectively capture local and global information. We acknowledge this limitation of the MSFEF module, which will be discussed in the conclusion section.)

The output of each partial convolution layer undergoes element-wise multiplication with its corresponding learnable weight tensor of size 1x1xC (Figure 2). Subsequently, the three resulting tensors are summed together, representing the fusion step. By utilizing the learnable weight tensors, we obtain a weighted combination of the output feature maps from the three partial convolution layers in the MSFEF module. This fused value is then multiplied by the updated mask obtained from the 3×3 partial convolution layer, and this becomes the final output of our MSFEF module. TThe reason we do this is because we want the output feature map to be generated by convolution operation around only those positions of the input feature map, where the partial convolution operation could be performed by all the three partial convolution layers in the MSFEF module. This will ensure the appropriate and **simulatenous** combination of global and local semantic information in the inpainting of the missing pixels. Finally, the updated mask from the 3×3 partial convolution layer is taken as the final updated mask of our MSFEF module.

Our hypothesis posited that the MSFEF module, functioning as described above, would effectively incorporate both local and global information in the inpainting of missing pixels. In the subsequent section, we will present experimental evidence supporting the validation of our hypothesis.

3.3 Loss functions

The loss functions used to train our model, as mentioned before, would be MSE loss, Perceptual loss and Style loss. These functions are as follows:

1. **MSE Loss:** This measures the difference between the raw pixel values of the generated inpainting result and the ground truth image. Given the ground truth image Y_{gt}, the inpainting result Y_{out}, and the original binary mask M, the MSE Loss is composed of two sub-losses:
   - $L_{avail} = \frac{1}{N_{gt}}(||M \circ (Y_{out} - Y_{gt})||_2)^2$
   - $L_{missing} = \frac{1}{N_{gt}}(||(1 - M) \circ (Y_{out} - Y_{gt})||_2)^2$

   Here N_{gt} denotes the total number of pixels in the ground truth (and obviously the generated image as well) given by $C * H * W$, where C, H, W denote the number of channels, height and width of the ground truth (and the generated image). L_{avail} and $L_{missing}$ denote the per-pixel MSE losses calculated separately for the available and missing regions (as per the given binary mask M and the ground truth Y_{gt}).
2. **Perceptual Loss:** The Perceptual Loss, (first introduced by Gatys et al. [5]) that we use for training our model is as follows:

$$L_{percept} = \sum_{l=1}^{L} \frac{||\Phi_l^{Y_{out}} - \Phi_l^{Y_{gt}}||_1}{N_{\Phi_l^{Y_{gt}}}} + \sum_{l=1}^{L} \frac{||\Phi_l^{Y_{comp}} - \Phi_l^{Y_{gt}}||_1}{N_{\Phi_l^{Y_{gt}}}} \tag{3}$$

   Here Y_{comp} represents the generated output image of our model Y_{out} but with the available pixels set directly to their corresponding ground-truth pixel values. From the above formulation, we find that perceptual loss is not the direct raw pixel-wise L1 loss between Y_{comp} and Y_{gt} and between Y_{out} and Y_{gt}. Instead, it is the L1-loss we compute pixel loss between Y_{out}, Y_{comp}, and Y_{gt}, after projecting them onto a higher-level latent feature space using a VGG-16 model[32] pre-trained on the ImageNet data.

   In Eq. 3, $\Phi_l^{Y_*}$ denotes the output feature map of the l-th selected layer among the L selected layers of VGG-16 given original input Y_*. $N_{\Phi_l^{Y_{gt}}}$ is the number of elements in the tensor $\Phi_l^{Y_{gt}}$. For example, in our implementation, we consider the outputs from the first, second and third pooling layers of VGG-16.
3. **Style Loss:** Style loss, also introduced in [5], is quite similar to the perceptual loss. However, in style loss, an autocorrelation matrix, also known as the Gram matrix is first computed for each feature map $\Phi_l^{Y_*}$, before the L_1 loss is computed. The Style loss is finally given as follows:

$$L_{style} = L_{style,out} + L_{style,comp} \tag{4}$$

   Here $L_{style,out}$ and $L_{style,comp}$ are defined as follows:
   - $L_{style,out} = \sum_{l=1}^{L} \frac{1}{C_p.C_p} ||K_p((\Phi_l^{Y_{out}})^T \Phi_l^{Y_{out}} - (\Phi_l^{Y_{gt}})^T \Phi_l^{Y_{gt}})||_1$
   - $L_{style,comp} = \sum_{l=1}^{L} \frac{1}{C_p.C_p} ||K_p((\Phi_l^{Y_{comp}})^T \Phi_l^{Y_{comp}} - (\Phi_l^{Y_{gt}})^T \Phi_l^{Y_{gt}})||_1$

   Here $L_{style,out}$ and $L_{style,comp}$ represent the constituent style loss terms for Y_{out} and Y_{comp} respectively, similar to what we saw perceptual loss. The calculations performed in the expression of $L_{style,out}$ and $L_{style,comp}$ as shown above assume that the feature maps $\Phi_l^{Y_*}$ are of shape $(H_l * W_l) * C_l$, resulting in a $C_l * C_l$ autocorrelation matrix. K_l is the normalizing factor $(1/((H_l * W_l * C_l))$ for the l-th selected layer among the L selected layers of VGG-16.

Finally, we calculate our total training loss as follows:

$$L_{total} = w_1 L_{avail} + w_2 L_{missing} + w_3 L_{percept} + w_4 L_{style} \tag{5}$$

We will treat w_1, w_2, w_3, w_4 as a part of the hyper-parameter set for our model.

4 Experiments and Results

In this section, we will discuss the experiments we have run, including hyper-parameter tuning, generated results and ablation studies.

4.1 Datasets

To provide quantitative and qualitative insights into the performance of our inpainting model, we train and evaluate the performance of our model on the following datasets:

- **Helen:** The Helen dataset [2] consists of a total of 2330 images with an average resolution of 400×400 pixels, among which 2000 images are for training and 330 images are for testing. The dataset consists of images with highly accurate, detailed, and consistent annotations of facial components, generated through manually-annotated contours along eyes, eyebrows, nose, lips and jawline. The Helen dataset serves as our primary dataset, and we use it for performing both qualitative analysis, as well as the ablation study of our model.
- **Stanford Cars:** The Stanford Cars dataset [3] contains 16,185 images of 196 classes of cars, with an average image size of 600×400. The data has been originally partitioned into 8,144 training images and 8,041 testing images, and each class has been split between the training and testing set roughly in a 50-50 split. The individual car classes are typically denoted by its Make, Model, and Year of release. In this report, we use the Stanford Cars dataset only for performing the qualitative analysis of our inpainting model.
- **LSUN Church Outdoor:** The LSUN dataset [4] consists of 10 scene categories, including dining room, bedroom, chicken, outdoor church, and more. The training data for each category contains a substantial number of images, ranging from approximately 120,000 to 3,000,000. The validation data consists of 300 images, and there are 1000 images available for testing in each category.

  For training our inpainting model, we specifically utilize a subset of the LSUN dataset i.e. the outdoor church category, which contains over 126,000 training images and 300 testing images. This subset is used for conducting qualitative analysis of the model's performance.

4.2 Training and Validation

1. Batch Size: The batch size is chosen to be 32, since a lower batch size leads to better training (given time constraints, the batch size could not have been chosen any lower).
2. Mask size: The size of the mask during training has been taken to be between (0.1*height of image, 0.25*height of image) Taking variations in the mask heights would lead to a better model, but that would require more epochs for convergence which would require more time and resources.
3. Loss function: Apart from the MSE loss, we have also incorporated a perceptual loss and style loss in our training. Gatys et al.'s work on neural style transfer involves two components: perceptual loss and style loss. Perceptual Loss measures the similarity between the generated image and a target image in terms of content. It compares the feature representations of both images at a specific layer of a pre-trained convolutional neural network (CNN). The goal is to ensure that the generated image captures the same high-level content and structure as the target image. Style Loss, on the other hand, focuses on capturing the artistic style of a reference image and transferring it to the generated image. It is based on the observation that style can be characterized by correlations between different feature maps in a CNN. The style loss measures the discrepancy between the correlations of feature maps in the generated image and those in the reference image. This is done by comparing the Gram matrices, which represent style information, at multiple layers of the CNN. The mathematical formulation of these losses is explained in the previous section.
4. Optimizer: The Adam optimizer is chosen for this network, which is better and faster than other commonly used optimizers (such as SGD). It takes the exponential weight average of the gradients which helps it converge faster to the minima of the loss function and stabilize the training process. Weight decay is also incorporated for regularization.
5. Learning Rate: The learning rate is derived from a plateau function which adjusts itself if no improvement is seen for a certain number of epochs during training. The initial learning rate is chosen to be 0.003.
6. Coefficients for training loss: The coefficients are chosen to be the same as obtained by the authors in the original partial convolution paper [1].

$$w_1 = 1, w_2 = 6, w_3 = 0.01, w_4 = 120$$

7. Epochs: The model is trained for 120 epochs, and while we see an appreciable decline till 100 epochs, the training slows down after that.

The metrics tracked during validation are SSIM, L2 Pixel loss, LPIPS loss and PSNR. Structural Similarity Index Measure (SSIM) is utilized to estimate the similarity between two images. L2 per-pixel loss function is used as a metric for understanding differences between images on a pixel level. The Peak Signal to Noise Ratio (PSNR) is a metric that indicates low noise in images when it is high, implying that the generated and base images are more similar. To assess the perceptual similarity between two images, the Learned Perceptual Image Patch Similarity (LPIPS) is employed. LPIPS calculates the similarity between the activations of two image patches using a predefined network.

4.3 Results

Below are examples generated by our inpainting model. The leftmost image represents the ground truth, the center image shows the input to the model, and the rightmost image displays the generated output. The model demonstrates good performance in generating images, effectively capturing the context, texture, and color.

Figure 3: *Helen*

Figure 4: *LSUN Church Outdoor*

Figure 5: *Stanford Cars*

4.4 Ablation Study

The goal of the ablation study is to highlight the effect of incorporating all loss components (instead of a subset) in training and more importantly, to demonstrate the effectiveness of using the MSFEF module over the use of ordinary single partial convolution layers. The losses incorporated during training, as we know, are the Mean Squared Error (MSE), Perceptual and Style losses. We aim to find the effect of omitting each of the loss functions: Peak Signal-to-Noise Ratio, Structural Similarity Index Measure (SSIM), Learned Perceptual Image Patch Similarity (LPIPS) loss and L2 Pixel-wise loss (MSE), and the MSFEF module on the validation metrics. The goal is to achieve the lowest L2 loss, highest SSIM, highest PSNR and lowest LPIPS values for our approach. Table 1 (next page) is

the ablation study done on the Helen dataset, where we have varied the hole-to-image ratios between 0.01 to 0.5.

Metric | Mask-to-Image ratio	**[0.01, 0.1)**	**[0.1, 0.2)**	**[0.2, 0.3)**	**[0.3, 0.4)**	**[0.4, 0.5)**
L2 (All loss + MSFEF)	0.0014	**0.0046**	**0.0086**	**0.0140**	**0.0183**
L2 (No MSE + MSFEF)	0.0034	0.0097	0.0226	0.0312	0.0407
L2 (No Perceptual + MSFEF)	0.0024	0.0086	0.0157	0.0231	0.0274
L2 (No Style + MSFEF)	**0.0013**	0.0053	0.0106	0.0154	0.0260
L2 (No perceptual, no style + MSFEF)	0.0026	0.0055	0.0102	0.0170	0.0235
L2 (All loss + No MSFEF)	0.0035	0.0075	0.0143	0.0202	0.0270
SSIM (All loss + MSFEF)	**0.9525**	0.8806	**0.8146**	0.7272	**0.6876**
SSIM (No MSE + MSFEF)	0.9352	0.8464	0.7364	0.6691	0.6225
SSIM (No Perceptual + MSFEF)	0.8979	0.8234	0.7424	0.6818	0.6445
SSIM (No Style + MSFEF)	0.9467	**0.8821**	0.8086	**0.7511**	0.6741
SSIM (No perceptual, no style + MSFEF)	0.8819	0.8305	0.7690	0.7059	0.6701
SSIM (All loss + No MSFEF)	0.8871	0.8284	0.7276	0.6566	0.6078
PSNR (All loss + MSFEF)	28.3496	23.4008	**20.6515**	**18.5389**	**17.3642**
PSNR (No MSE + MSFEF)	24.6991	20.1296	16.4730	14.9620	13.9297
PSNR (No Perceptual + MSFEF)	26.1351	20.6458	18.0514	16.3660	15.6836
PSNR (No Style + MSFEF)	**28.9544**	22.7670	19.7333	18.1414	15.8722
PSNR (No perceptual, no style + MSFEF)	25.8145	22.7670	19.7333	18.1414	15.8722
PSNR (All loss + No MSFEF)	25.8145	**23.8689**	19.9081	17.6889	16.3764
LPIPS (All loss + MSFEF)	**0.1549**	**0.2016**	**0.2446**	0.3097	**0.3441**
LPIPS (No MSE + MSFEF)	0.1612	0.2385	0.3171	0.3632	0.3917
LPIPS (No Perceptual + MSFEF)	0.1907	0.2472	0.3328	0.3777	0.4123
LPIPS (No Style + MSFEF)	0.1685	0.2119	0.2588	**0.2993**	0.3473
LPIPS (No perceptual, no style + MSFEF)	0.2597	0.2851	0.3229	0.3627	0.4037
LPIPS (All loss + No MSFEF)	0.1850	0.2327	0.3097	0.3546	0.3802

Table 1: Ablation Study

Observations: From the values highlighted in bold, which indicate the best value we have obtained for a particular metric and mask-to-image ratio, we observe that for all the metrics, for most of the mask-to-image ratios, our approach (All loss + MSFEF) achieves the best values, indicating that our approach indeed works! Even if it's not the best value, it achieves the second-best value. If we consider the last row for each metric, where MSFEF is replaced by the Partial Convolution module, we observe that metrics are worse. So we can conclude from the ablation study that, when we incorporate all losses (L2, Perceptual, Style) and the MSFEF module in the network for training, we arrive at the best-performing model. And as we can see from our results as well, the inpainting of the masked images is done pretty impressively.

5 Conclusions and Future Work

Partial Convolution Layers have become a crucial component in deep learning-based inpainting methods. Recent approaches often use only a single layer per convolution stage, which is insufficient for images with significant missing portions or sparse details. This limitation arises from the straightforward implementation lacking effective integration of global and local information for inpainting missing pixels.

To address this, we propose the Multi-Scale Feature Extraction and Fusion (MSFEF) module, which appropriately combines global and local information. Experimental results demonstrate that incorporating the MSFEF module improves both quantitative and qualitative performance, effectively estimating missing pixel values by considering surrounding semantic information. Furthermore, the generated inpainting results (as shown in Figures 3, 4, 5) show that our inpainting method is able to incorporate both the local and global semantic information around the missing pixels to estimate their values.

An important consideration lies in appropriately adjusting the kernel sizes of the partial convolution layers within the MSFEF module, taking into account the resolution of the input image. In our case, we used kernel sizes of 7x7, 5x5, and 3x3 for capturing global and local information in 256x256 images. However, kernel size selection depends on the resolution, and choosing them properly is necessary for the effective integration of local and global information.

To further improve the inpainting model, techniques like Neural Architecture Search (NAS) can be explored to determine the optimal number of MSFEF layers and module configuration. Novel heuristics for kernel size determination can also be investigated to mitigate limitations and enhance overall performance.

References

[1] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 85–100, 2018.

[2] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, "Interactive facial feature localization," in *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III 12*, pp. 679–692, Springer, 2012.

[3] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

[4] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[6] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 417–424, 2000.

[7] T. F. Chan and J. Shen, "Nontexture inpainting by curvature-driven diffusions," *Journal of visual communication and image representation*, vol. 12, no. 4, pp. 436–449, 2001.

[8] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, 2009.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[10] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

[11] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5505–5514, 2018.

[12] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–14, 2017.

[13] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 1–17, 2018.

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

[15] Z. Wan, J. Zhang, D. Chen, and J. Liao, "High-fidelity pluralistic image completion with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4692–4701, 2021.

[16] C. Zheng, T. Cham, and J. Cai, "Pluralistic image completion," *CoRR*, vol. abs/1903.04227, 2019.

[17] L. Zhao, Q. Mo, S. Lin, Z. Wang, Z. Zuo, H. Chen, W. Xing, and D. Lu, "Uctgan: Diverse image inpainting based on unsupervised cross-space translation," pp. 5740–5749, 06 2020.

[18] S. Zhao, J. Cui, Y. Sheng, Y. Dong, X. Liang, E. I. Chang, and Y. Xu, "Large scale image completion via co-modulated generative adversarial networks," *CoRR*, vol. abs/2103.10428, 2021.

[19] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," *CoRR*, vol. abs/1912.04958, 2019.

[20] J. Peng, D. Liu, S. Xu, and H. Li, "Generating diverse structure for image inpainting with hierarchical VQ-VAE," *CoRR*, vol. abs/2103.10022, 2021.

[21] Z. Wan, J. Zhang, D. Chen, and J. Liao, "High-fidelity pluralistic image completion with transformers," *CoRR*, vol. abs/2103.14031, 2021.

[22] Y. Yu, F. Zhan, R. Wu, J. Pan, K. Cui, S. Lu, F. Ma, X. Xie, and C. Miao, "Diverse image inpainting with bidirectional and autoregressive transformers," *CoRR*, vol. abs/2104.12335, 2021.

[23] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or, "Encoding in style: a stylegan encoder for image-to-image translation," *CoRR*, vol. abs/2008.00951, 2020.

[24] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018.

[25] K. C. K. Chan, X. Wang, X. Xu, J. Gu, and C. C. Loy, "GLEAN: generative latent bank for large-factor image super-resolution," *CoRR*, vol. abs/2012.00739, 2020.

[26] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin, "PULSE: self-supervised photo upsampling via latent space exploration of generative models," *CoRR*, vol. abs/2003.03808, 2020.

[27] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Deep image prior," *CoRR*, vol. abs/1711.10925, 2017.

[28] M. Chen, S. Zang, Z. Ai, J. Chi, G. Yang, C. Chen, and T. Yu, "Rfa-net: Residual feature attention network for fine-grained image inpainting," *Engineering Applications of Artificial Intelligence*, vol. 119, p. 105814, 2023.

[29] W. Wang, J. Zhang, L. Niu, H. Ling, X. Yang, and L. Zhang, "Parallel multi-resolution fusion network for image inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14559–14568, 2021.

[30] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4471–4480, 2019.

[31] C. Xie, S. Liu, C. Li, M.-M. Cheng, W. Zuo, X. Liu, S. Wen, and E. Ding, "Image inpainting with learnable bidirectional attention maps," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8858–8867, 2019.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

Appendix

Module Name	Filter Size	Filters/Channels	Stride/Up Factor	Normalization	Nonlinearity
MSFEF 1	7x7, 5x5 and 3x3	64	2	IN	ReLU
MSFEF 2	7x7, 5x5 and 3x3	128	2	IN	ReLU
MSFEF 3	7x7, 5x5 and 3x3	256	2	IN	ReLU
MSFEF 4	7x7, 5x5 and 3x3	512	2	IN	ReLU
PConv1	3x3	512	2	IN	ReLU
PConv2	3x3	512	2	IN	ReLU
PConv3	3x3	512	2	IN	ReLU
PConv4	3x3	512	2	BN	ReLU
NearestUpSample1		512	2	-	-
Concat1(w/ PConv7)		512+512		-	-
PConv9	3x3	512	1	BN	LeakyReLU(0.2)
NearestUpSample2		512	2	-	-
Concat2(w/ PConv6)		512+512		-	-
PConv10	3x3	512	1	BN	LeakyReLU(0.2)
NearestUpSample3		512	2	-	-
Concat3(w/ PConv5)		512+512		-	-
PConv11	3x3	512	1	BN	LeakyReLU(0.2)
NearestUpSample4		512	2	-	-
Concat4(w/ PConv4)		512+512		-	
PConv12	3x3	512	1	BN	LeakyReLU(0.2)
NearestUpSample5		512	2	-	-
Concat5(w/ PConv3)		512+256		-	-
PConv13	3x3	256	1	BN	LeakyReLU(0.2)
NearestUpSample6		256	2	-	-
Concat6(w/ PConv2)		256+128		-	-
PConv14	3x3	128	1	BN	LeakyReLU(0.2)
NearestUpSample7		128	2	-	-
Concat7(w/ PConv1)		128+64		-	
PConv15	3x3	64	1	BN	LeakyReLU(0.2)
NearestUpSample8		64	2	-	-
Concat8(w/Input)		64+3		-	-
PConv16	3x3	3	1	-	-

Table 2: Model Architecture: MSFEF1-4 and PConv1-4 are part of the encoder module and the rest of the modules are part of the decoder. IN refers to Instance Norm and BN refers to Batch Norm