Unveiling the Invisible: Developing a UNET-based Algorithm for Image Inpainting

Jon Torgeir Grini
Department of Electrical and Computer Engineering
U09464662

Elias Buoe
Department of Electrical and Computer Engineering
U09431415

Abstract

This report presents an approach for image inpainting, which is the process of filling in missing or damaged parts of an image with plausible content. Our proposed approach is based on a UNET-like network with standard convolutions that takes the incomplete image as input and generates the complete image as output. The network architecture consists of an encoder network that extracts high-level features from the input image, a decoder network that generates the complete image from the features, and skip connections that allow the decoder to access low-level details from the encoder. The network is trained on images with missing regions and their corresponding ground-truth images using the COCO dataset. Image inpainting is a challenging task that requires the algorithm to understand the context and structure of the surrounding image regions and use this information to generate plausible content. Our approach aims to address this challenge by leveraging the benefits of the UNET architecture and using standard convolutions to learn the mapping between the incomplete image and its complete version. The proposed approach has potential applications in photo restoration, object removal, and video editing. In the related work section, we discuss the papers on unsupervised feature learning and inpainting irregularly shaped holes. Additionally, we review an article that explains the concept of image inpainting and deep learning-based approaches for inpainting. Our proposed approach shows promising results in low-texture image inpainting tasks and has the potential to be applied to a wide range of image restoration applications. Future work can focus on making it more effective for restoring high-texture images and further improving its performance.

1 Introduction

In this project, we aim to develop an image inpainting algorithm that can effectively restore images with missing or occluded regions. The motivation for this project comes from the fact that many images are prone to damage or loss due to various reasons, such as data corruption, compression, and physical damage. Inpainting these images can restore their visual quality and usefulness, and can be particularly valuable for historical or cultural artifacts that are irreplaceable.

Image inpainting is a challenging task as it requires the algorithm to understand the context and structure of the surrounding image regions and use this information to generate plausible content. Our approach to solving this problem is based on using a UNET like network with standard convolutions to learn the mapping between the incomplete image and its complete version.

Our proposed approach consists of a UNET-like network with standard convolutions that takes the incomplete image as input and generates the complete image as output. The network architecture consists of an encoder network that extracts high-level features from the input image, a decoder network that generates the complete image from the features, and skip connections that allow the decoder to access low-level details from the encoder. The network is trained on a dataset of images with missing regions and their corresponding ground-truth images using a pixel-wise loss function.

2 Related Work

As a starting point for our project, we read the papers "Context Encoders: Feature Learning by Inpainting" [1] and "Image inpainting for irregular holes using partial convolutions" [2], before finding the article [3] and related code [4]. The two papers and the article gives comprehensive descriptions of three well performing image inpainting methods, which will be discussed below. The latter one, i.e the work by Ayush Thakur and Sayak Paul [3], is what we based our work on.

The paper "Context Encoders: Feature Learning by Inpainting" [1] introduces a novel deep learning framework called "Context Encoders" for unsupervised feature learning. Context Encoders learn feature representations by completing or "inpainting" missing portions of an input image. Specifically, the encoder network takes an input image and encodes it into a lower-dimensional feature vector. This feature vector is then passed through a decoder network, which tries to reconstruct the original image by filling in the missing portions.

The authors demonstrate that this inpainting task forces the network to learn meaningful feature representations that capture high-level information about the input image, such as object shape and texture. They show that Context Encoders outperform several state-of-the-art unsupervised feature learning methods on several image classification tasks. Furthermore, the authors propose a new loss function called the "contextual loss" that measures the similarity between the reconstructed image and the original image, but only in the regions where the original image is available. This loss function encourages the network to learn features that are consistent with the context of the input image.

In the second paper, "Image inpainting for irregular holes using partial convolutions" [2], the authors focus on the problem of inpainting irregularly shaped holes in images, which is a challenging task that requires a sophisticated approach. The proposed method, a Partial Convolutional Neural Network (PCNN), is based on a modified version of the standard convolutional neural network (CNN) architecture. The main idea behind PCNN is to adapt the convolution operation to handle irregular holes by only using valid pixels for convolutional filters. The authors achieve this by introducing a mask that indicates which pixels in the input image are valid and which ones are not.

During training, the network learns to predict the missing pixels in the masked region by only using the valid pixels in the convolution operation. The authors also introduce a new loss function that encourages smoothness and consistency between the inpainted region and the surrounding pixels. The experimental results show that the proposed PCNN method outperforms state-of-the-art methods on several benchmark datasets, both in terms of quantitative metrics and visual quality.

The article "Introduction to image inpainting with deep learning" [3] explains the concept of image inpainting, the challenges involved, and various deep learning-based approaches used for inpainting. Furthermore, it introducing the basics of convolutional neural networks (CNNs) and then discusses various inpainting models such as Generative Adversarial Networks (GANs), Autoencoders, and Context Encoders, and provide an overview of the training process for each of these models.

3 Method

The implemented convolutional neural network is a modified U-Net architecture. U-Net is a popular convolutional neural network (CNN) used for various image segmentation and reconstruction tasks, including image inpainting, which is the task of filling in missing or damaged parts of an image. The architecture figure is visualized in Appendix A

The U-Net architecture is characterized by a contracting path and an expanding path, which allows for a high-level understanding of the image while maintaining precise localization information. The contracting path learns to encode the contextual information of the input image, while the expanding path generates the missing or damaged parts of the image based on the learned contextual information.

The contracting path consists of several convolutional and max-pooling layers, which reduce the spatial resolution of the input image while increasing the number of feature channels. On the other hand, the expanding path consists of convolutional and upsampling layers, which gradually increase the spatial resolution of the feature maps while reducing the number of feature channels. The final output of the U-Net is an inpainted image based on the input image.

The contracting path in the modified U-Net architecture consists of four blocks, each consisting of two convolutional layers with 3x3 filters and ReLU activation functions, followed by a max-pooling layer with a 2x2 window. The number of filters in the first block of the contracting path is 32, and it doubles in each subsequent block. Filters are like "feature detectors" that are applied to the input image to detect specific patterns. This architecture is similar to the original U-Net architecture [5], but with fewer blocks and fewer filters per block.

In the expanding path, the feature maps are upsampled to the original spatial dimensions using transpose convolution layers (also known as deconvolution layers), which increase the spatial dimensions of the feature maps while reducing the number of channels. The upsampled feature maps are then concatenated with the corresponding feature maps from the contracting path using the concatenate layer, which combines the feature maps along the channel dimension. In each block, the concatenated feature map is passed through two convolutional layers with decreasing number of filters, which generate the missing or damaged parts of the image based on the contextual information learned in the contracting path. ReLU activation functions are used after each convolutional layer in the expanding path.

The skip connections between the contracting and expanding paths (i.e., the concatenation layers) help preserve the high-resolution details in the input image that may be lost during downsampling in the contracting path.

The output layer of the modified U-Net architecture consists of a convolutional layer with a 3x3 filter and a sigmoid activation function, which produces the output image with the same size as the input image. The sigmoid activation function is used to ensure that the output image has pixel values between 0 and 1, which represents the intensity of each color channel. The output image is the result of combining the original image with the generated missing or damaged parts.

The modified U-Net architecture has some strengths compared to other architectures. Firstly, the U-Net architecture is designed to preserve spatial information while learning high-level features, making it particularly suitable for tasks that require precise localization information. This is achieved through the use of skip connections between the contracting and expanding paths, which allow the network to access low-level features from earlier layers. Secondly, the modified U-Net architecture has a relatively small number of parameters compared to other state-of-the-art models for image inpainting, making it easier to train and deploy on resource-limited devices.

However, the modified U-Net architecture also has some limitations. Firstly, it might not be suitable for images with complex structures, such as images with large occlusions or multiple missing regions, as it relies on a simple encoder-decoder structure. Secondly, the modified U-Net architecture may struggle to inpaint images with fine details, such as textures or small objects, as it reduces the spatial resolution of the input image during the contracting path.

4 Experiments

This section includes a description of the dataset used to train and test the proposed network. Furthermore, we describe how we used random erasing to augment and modify the images to make them suitable for the image inpainting task. Finally, we present the obtained results.

4.1 Dataset

We utilized the Common Objects in Context (COCO) dataset [6] for our image inpainting task due to several reasons. Firstly, the dataset contains diverse images that include various objects, scenes, and backgrounds, making it a good option to train and evaluate the performance of the inpainting model. Additionally, the dataset comprises 330,000 images, providing sufficient data for training deep neural networks that generate high-quality inpainted images. Moreover, COCO is an object-centric dataset that focuses on common objects and their interactions, which is beneficial for inpainting tasks

that require filling in missing parts of objects. Lastly, COCO also provides annotations for each image, including object bounding boxes, categories, and segments which could be used to create more structured input for the inpainting model.

In our project, we downloaded the COCO training 2017 dataset in jpg-format. We then preprocessed the data by loading the images into Python and then resizing all images to a smaller size of 32x32 pixels. This allowed us to reduce the memory requirements during training, speed up the training process, and make the model more computationally efficient. However, we were aware that this may have resulted in a loss of detail and potentially degraded the accuracy of the model. Additionally, we removed all grayscale images from the dataset. This was done as grayscale images contain only shades of gray and no color information, and may not be representative of the type of images we were interested in analyzing. By removing these images, we ensured that our dataset only contained the types of images that we were interested in analyzing.

We used a train set size of approximately 54,000 samples and a test set size of approximately 5,000 samples to train and test our model. The reason we used a smaller subset of the training set was to reduce the training time required for our model. We experienced that training deep learning models can be a computationally expensive process, and by using a smaller subset of the training set, we were able to reduce the time required to train our model significantly.

While using the whole training set may have resulted in a more accurate model, we were able to achieve satisfactory results with the smaller subset of the data we used. Ultimately, the decision to use a smaller subset of the training set was a trade-off between model accuracy and training time, and allowed us to complete our project in a timely manner while still achieving good results.

4.2 Data Augmentation

In the context of data augmentation, we used random erasing to augment our data. This involves randomly erasing (i.e., setting to zero) a rectangular region of an image during training, which can encourage the model to learn more robust features and help to prevent overfitting.

Random erasing can be particularly useful in the context of image inpainting, as by randomly erasing rectangular regions of the image during training, the model can learn to fill in missing data more effectively and become more robust to different types of missing data.

Random erasing can also make the model more robust to specific types of missing data. For example, if the missing data is in the form of rectangular regions, the model will be better equipped to handle this type of missing data if it has been trained on images that have had similar rectangular regions erased. However, this is also a limitation, as a model trained on missing rectangular regions might perform worse on other shapes of missing information.

4.3 Results

Training our convolutional neural network on the COCO dataset for 10 epochs, we obtained results as shown in Figure 1. These results indicates that the network can restore missing parts of images accurately, particularly in areas with low texture. In these cases, it is rather difficult in the output image to identify the location of the previously missing region. For instance, the image in the lower left corner shows the restored color of the sky, making it difficult to pinpoint the exact location of the missing part.

However, in textured areas, the network faced difficulties in generating precise reconstructions, resulting in restored parts lacking detailed information despite having the correct color scheme. An example is the image in the lower right corner, where the output image's color scheme matches the original image, but the details seem to have undergone some Gaussian blur. Compared to the original image, some information is lost in the output image.

Overall, our experiments indicate that the network can be a useful tool for image inpainting, but its performance is dependent on the complexity of the image content.

Figure 2 displays the loss function of the training and validation sets. The initial 3 epochs indicate a significant reduction in the loss function, suggesting that the model improves in reconstructing missing parts. After that, we observe a stagnation in the development of the loss function, indicating a lack of substantial improvement in the model's performance. Despite this, there's no evidence of

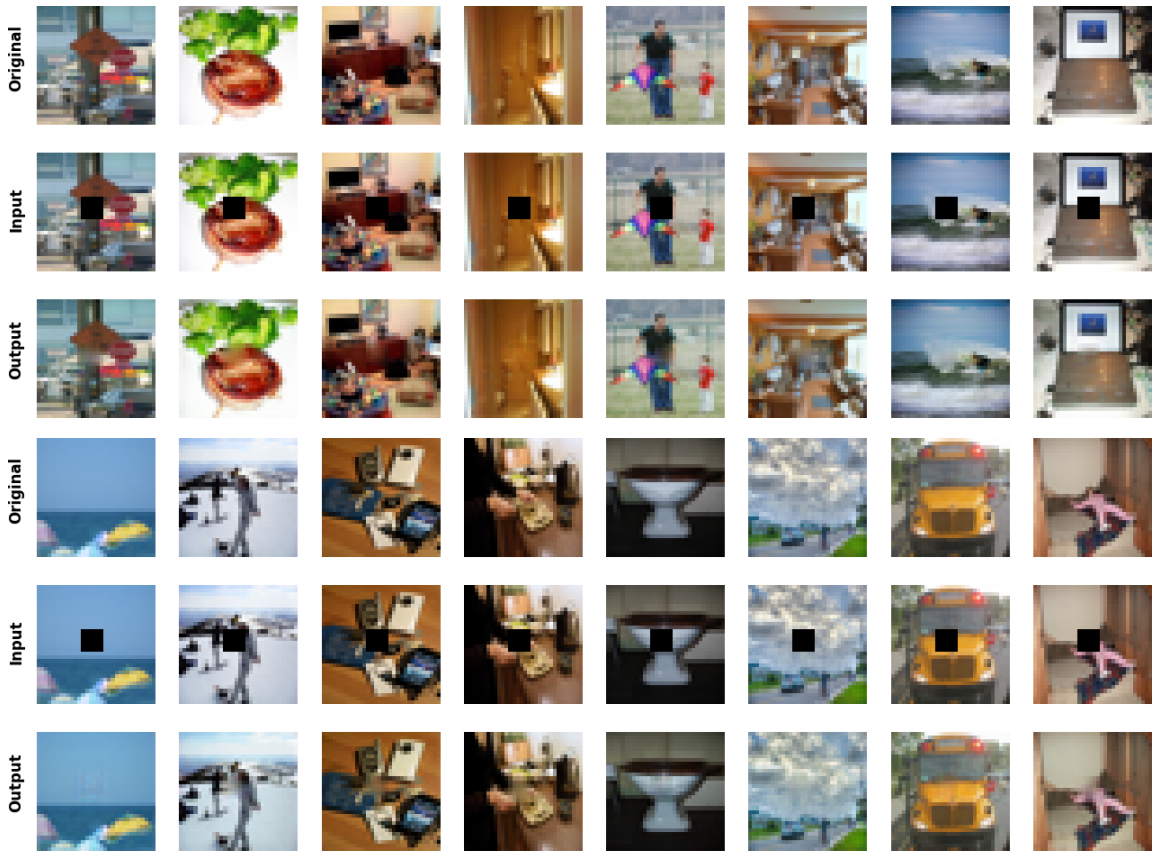

Figure 1: Results of inpainting after 10 epochs

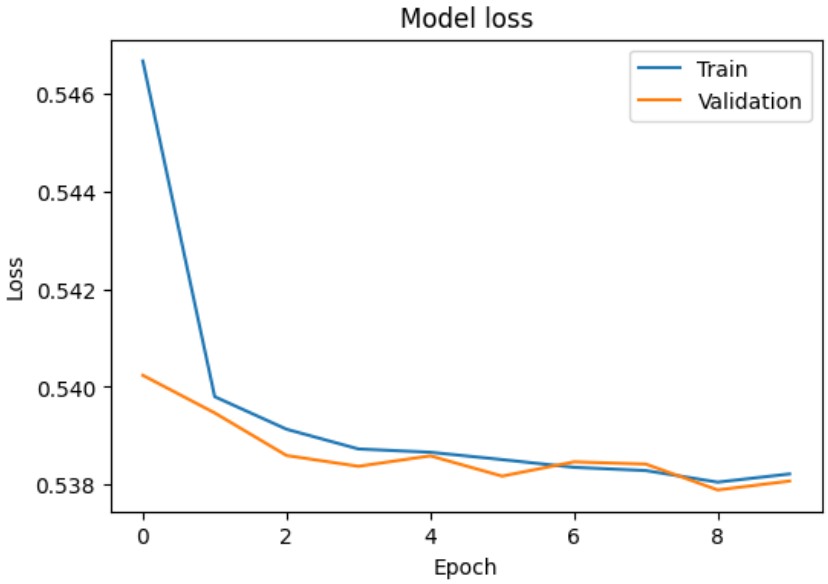

Figure 2: Training and Validation Accuracy

overfitting as the loss function of the validation set remains relatively stable. We could have reduced the computational cost by decreasing the number of epochs while still obtaining satisfactory results.

References

[1] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., & Efros, A. A, "Context Encoders: Feature Learning by Inpainting", 2016, https://arxiv.org/abs/1604.07379 (3/10/2023)

[2] Liu, G., Reda, F. A., Shih, K. J., Wang, T. C., Tao, A., & Catanzaro, B., "Image inpainting for irregular holes using partial convolutions", 2018, https://arxiv.org/abs/1804.07723 (3/12/2023)

[3] Thakur, A. & Paul, S., "Introduction to image inpainting with deep learning", 2020, https://wandb.ai/site/articles/introduction-to-image-inpainting-with-deep-learning (3/14/2023)

[4] Thakur, A. & Paul, S., "GitHub: Introduction to image inpainting with deep learning", 2020, https://github.com/ayulockin/deepimageinpainting (3/14/2023)

[5] LMB, University of Freiburg, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2023, https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/ (3/16/2023)

[6] Petrosyan, Tigran. "Introduction to the COCO Dataset", 2021, https://opencv.org/introduction-to-the-coco-dataset/

A Architecture

Figure 3: Architecture [3]