TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

A

PROJECT REPORT

ON

IMAGE REPROCESSING OF OCCLUDED OBJECT

SUBMITTED BY:

AAKRIT DONGOL (PUL077BCT002)

BIRAJ KUMAR KARANJIT (PUL077BCT020)

KRISHALA PRAJAPATI (PUL077BCT038)

SUBMITTED TO:

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

10th March 2024

Copyright

Acknowledgments

We would like to express our profound gratitude and heartfelt appreciation to our esteemed project supervisor, Dr.Sanjib Prasad Pandey, Ph.D., for his invaluable guidance, unwavering support, and expert insights throughout the conceptualization and planning stages of this research endeavor.

Additionally, We extend our sincere thanks to the Department of Electronics and Computer Engineering at Pulchowk Campus, IOE for providing the necessary resources, facilities, and conducive environment for conducting this research. The collaborative atmosphere and scholarly interactions within the department have enriched our academic journey and contributed significantly to the progress of this project.

We are also grateful to our peers and mentors who have offered their encouragement, constructive critiques, and intellectual engagement, fostering a stimulating academic environment conducive to innovation and excellence.

We recognize and appreciate the collective efforts, insights, and contributions of everyone involved in this project, directly or indirectly. Your support and collaboration have been pivotal, and We are truly honored and privileged to embark on this scholarly endeavor under the guidance of Dr.Sanjib Prasad Pandey.

Abstract

The presence of occlusions in images often hampers their interpretability and utility in various applications. In this project, we aim to address the challenge of image reprocessing for occluded images using deep learning inpainting techniques. Our work focuses on employing neural networks specifically designed for inpainting tasks to restore and enhance the clarity of occluded images.

We propose a novel approach that uses convolutional neural networks (CNNs) to learn and reconstruct missing or occluded regions within images. By using a deep learning-based inpainting model, our method aims for accurately restoring occluded regions, thereby recovering the visual information lost due to occlusions.

Keywords: *Occluded images, Image reprocessing, Deep learning, Inpainting methods, Convolutional neural networks, Image restoration*

Contents

List of Figures

List of Abbreviations

CNN	Convolutional Neural Network
AI	Artificial Intelligence
GPU	Graphics Processing Unit
GUI	Graphical User Interface

1. Introduction

Image reprocessing of occluded object is the project related to the filed of image processing and computer vision.

1.1 Background

In the world of computer vision and image processing, the accurate identification and interpretation of objects within images play a pivotal role in diverse applications, ranging from autonomous systems to surveillance. However, the presence of occlusion, where objects are partially or entirely obscured by other entities or environmental factors, gives rise to various challenges. Overcoming occlusion is crucial for the successful deployment of technologies.So, this project emerges from the necessity to address this challenge and enhance the effectiveness of object recognition in complex visual scenarios.

Traditional image processing methods often struggle to effectively restore occluded regions, leading to a demand for advanced techniques. Techniques such as image inpainting, which involves predicting and filling in missing or occluded parts of an image, have provided good result in this kind of project. The use of partial CNN has also improved the inpainting feature of this kind of project. So, to overcome the problem of occluded images we plan to use the partial CNN technique in this project.

1.2 Problem statements

Occlusion introduces complexities in object recognition systems, leading to decreased accuracy and reliability. Existing technologies often struggle to provide precise identification and analysis when objects are hidden or partially obstructed by environmental elements, other objects, or dynamic changes in the scene. Also the object reconstructed using existing technologies are not clear enough and also do not blend with the image. These lead to the the need for the development of better and innovative solution for overcoming these limitations.

1.3 Objectives

To develop and implement a deep neural network model for manual occluded object selection, and integrate the models for effective occlusion removal in image reconstruction project.

1.4 Scope

1.4.1 Algorithmic Innovation:

Developing novel algorithms to accurately process occluded regions within images and generate almost similar image removing occluded object.

1.4.2 Integration of Deep Learning:

Exploring the integration of deep learning models to enhance the system's ability to adapt and generalize in the presence of various occlusion scenarios.

1.4.3 Real-World Applicability:

Ensuring that the developed technologies and methodologies are applicable across various domains, making a significant impact on fields such as healthcare, security, and industrial automation.

1.4.4 User Friendly Interface:

developing user friendly interface so that user does not face any difficulty while selecting the occluded object.

2. Literature Review

Image inpainting refers to the process of filling/completing missing regions across images with an estimated prediction according to the surrounding pixel information. The main objective of image inpainting is to produce completed recovered images in an unnoticeable manner to the human visual system. Image inpainting has various applications, such as image restoration, image editing , removal of the unwanted object, image denoising, and much more.

There are various algorithms for image inpainting.

1. Traditional Inpainting
2. Deep Learning Approach of Image Inpainting

2.1 Related work

2.1.1 AI Playground by Nvidia Research

AI Playground is an Image Inpainting project that allows you to edit images with image inpainting. Useing the power of Nvidia GPUs and deep learning algorithms, this project can replace any portion of the image.

2.2 Related theory

The Theories related to our project are discussed below:

2.2.1 Image Inpainting

Image Inpainting is the task of reconstructing missing region in an image. It is an important problem in computer vision and can be used to remove occlusion by inpainting the occluding part of the image.

fig. Pictures showing image Inpainting to remove an object

2.2.2 Neural Network

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. Just like human brain, Neural Network consists of a network of neurons(artificial neurons). A simple Neural Network consists of 3 layers :

- Input Layer : Information from the outside world enters the artificial neural network from the input layer.
- Hidden Layer : Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.
- Output Layer : The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0.

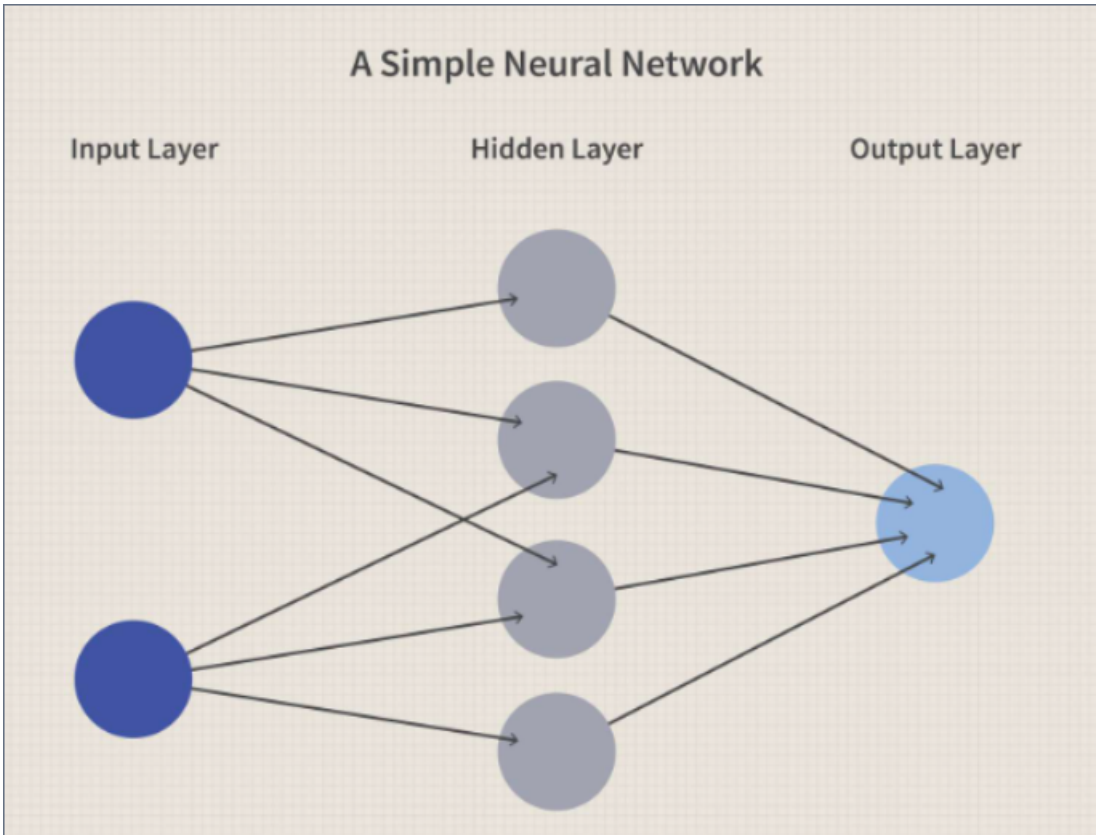

fig. Diagram of Simple Neural Network

A number, called weight, represents the connections between one node and another. The weight is a positive number if one node excites another, or negative if one node suppresses the other. Nodes with higher weight values have more influence on the other nodes.

2.2.3 Deep Learning

Deep learning is a method in artificial intelligence (AI) that teaches computers to process data and solve problems in a way that is inspired by the human brain. This approach uses multi-layered Neural Network.

2.2.4 Convolutional Network

Convolutional Neural Network (CNN) is a type of Deep neural network commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

It consists of 3 layers :

- Convolutional layer : This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred to as feature maps.
- Pooling Layer : This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory
- Fully connected layer : It takes the input from the previous layer and computes the final classification or regression task. Within this layer, all nodes from one layer are connected to every nodes of next layer.

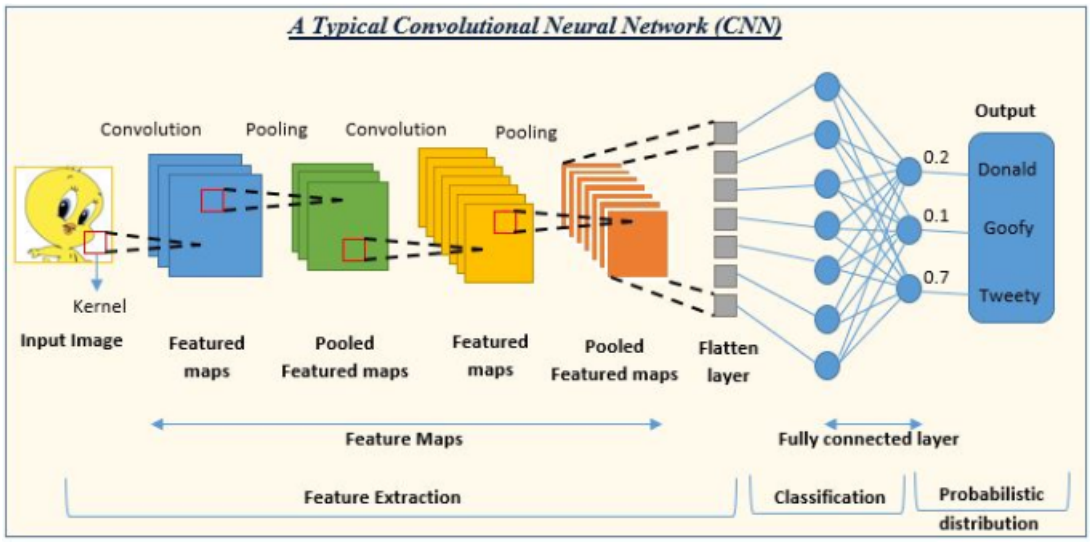

fig. Diagram of Convolutional Neural Network

2.2.5 U-Net Architecture

U-Net is a widely used deep learning architecture that was first introduced in the "U-Net: Convolutional Networks for Biomedical Image Segmentation" paper.

The architecture of U-Net network consists of following parts :

1)Encoder :

Encoder part involves feature extraction of the image. Each Encoder layer involves Convolution Layer followed ReLU activation function. The output of each encoder layer is called feature map of that layer.

2)Decoder:

Decoder part involves upsampling, concatenation and transpose convolution of the abstract representation of image followed by some activation function. In our project, we used leaky ReLU function with negative slope = 0.2.

3)Skip Connections:

Skip connections help to concatenate the high level feature maps of corresponding encoder layer with the upsampled image in the decoding layer.

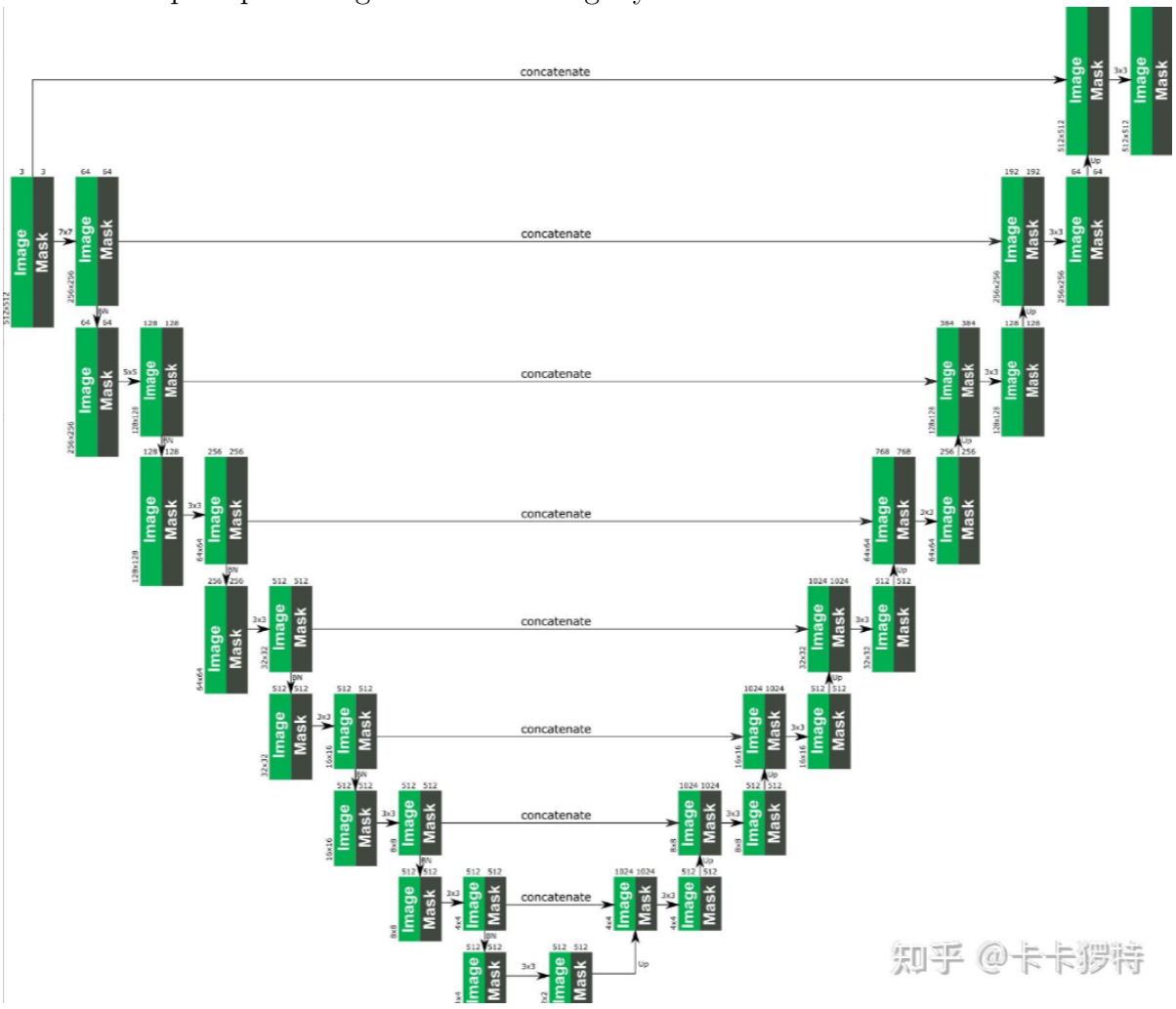

fig. Diagram of U-Net architecture

2.2.6 Partial Convolutional Inpainting

Partial Convolution is a key concept we use to image inpainting. We replace all the convolutional layers with partial convolution. Unlike the tradtional Convolutional Neural Network, Partial Contional Network determines whether a pixel is included in the convolution depending on the mask value. In normal convolution, the input (X) is multiplied by the weight (or kernel in CNN)(W). In this case, the missing pixels are also used for convolution, resulting in poor image quality. In Partial Convolutions, only the pixels, which are not the part of the region to be inpainted(pixel values of 255 in PIL Image format or 1 in tensor format), are used for convolution, greatly improving the image quality.

The Partial Convolution Layer consists of following formula :

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

fig. Formula for Partial Convolution of Image

As we can see from the formula, we are performing the convolution operation only with the region, whose corresponding mask position's pixel values are valid or unmasked region or important region for us($sum(M) > 0$, sum of all the pixels of the corresponding mask pixels is positive which ensures there is atleast one important feature for inpainting).

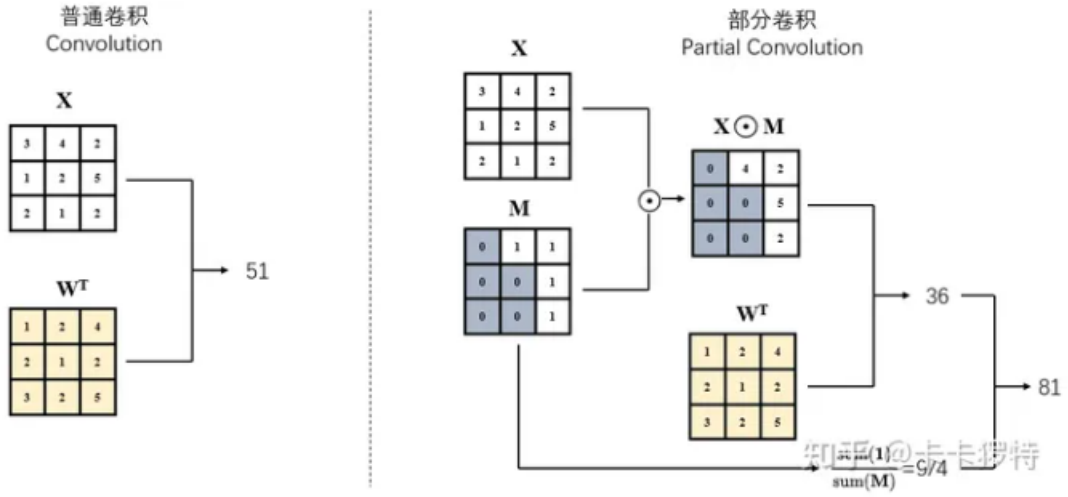

fig. Difference between Convolution and Partial Convolution

After the partial convolution, the next step is mask updation. In each layer, after performing Partial Convolution on the image to be inpainted, we update our mask because some invalid pixels in our original image(or image to be inpainted) are filled which can be useful for model to complete the missing regions of the original image.

$$m' = \begin{cases} 1, & \text{if } \text{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases}$$

fig. Formula for Mask Update

As we can see from the formula, if the sum of all the pixels of mask corresponding to the kernel position is greater than 0, we set the pixel value as 1, else we set it to 0. In this way, we update our mask in each layer of the model.

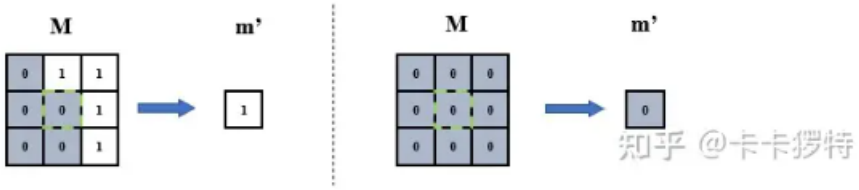

fig. Mask Update Process

3. Methodology

This section outlines a comprehensive methodology for addressing the challenge of removing occlusion from images. Utilizing Partial Convolutional Neural Networks (Partial CNN), our approach is structured to progress systematically from initial feasibility assessments to final deployment.

3.1 Feasibility Study

Based on different reports and research paper we studied,we figured out that using partial CNN method would be better for occlusion removal of the images. The necessary technology, tools and algorithms (Pytorch, Loss calculation algorithm, etc) for partial CNN method was easy to access but learning to use these tools were quite difficult task as we didn't have enough time to study about all of them in depth. Also developing the algorithm for the model was not that difficult as we followed the algorithm of the references used but we faced difficulty while training a model as we didn't have enough GPU in our CPU for training purpose.Overall, this project was quite challenging for us.

3.2 Requirement Analysis

We carefully identified what exactly we want to achieve with the project, what kind of data we need to collect, and any limitations or challenges we might face. We also analyzed about the functionality that our project should consist and what services should it provide to the user. As per our research some of the important functional and non-functional requirements are listed below:

3.2.1 Functional Requirements

User Interface

The user interface should be use friendly and it should clearly display the navigations and instructions for the user and it should be simple so that user can easily use the program. The selected image and output image should also be displayed clearly.

Mask Drawing

The developed system should allow the users to draw on the displayed image manually using the mouse at the area they want for inpainting and the drawing of mask should be smooth and user friendly.

Binary Mask Generation

After the user has drawn the mask on the selected image, the system should create the binary mask representing the area selected by user or based on mask drawn by user.

Image Inpainting

The system should be able to remove the occluded object and should reconstruct that area after removal of object.

Output Saving

After the system has performed the inpainting on the given image, the output produced by system should be saved in the device.

3.2.2 Non-functional Requirements

Performance

The image inpainting process should be efficient and the output should be accurate and it should support images with different resolutions.

Security

The uploaded image should be handled securely to maintain the user privacy.

Usability

The UI should be user-friendly, ensuring that users can easily navigate and interact with the system.

Training Data Diversity

The model should be trained on diverse data set, ensuring its ability to perform in different inpainting scenarios.

3.3 Data Collection and Pre-processing

The type of dataset required for this project were first identified and then we collected the dataset(occluded images) from the internet and then we created the binary mask for those dataset using "openCV".

After the collection of all the required data we converted those data into the form suitable for feeding into the Partial CNN model for training and testing. The data were converted into Pytorch tensor form using the inbuilt pytorch function ToTensor() and after conversion into tensor, it was then resized into 512x512 size.

```
# Data Transformation
img_tf = transforms.Compose([
            transforms.ToTensor(),
            transforms.Resize(size = (512,512))
            ])


mask_tf = transforms.Compose([
            transforms.ToTensor(),
            transforms.Resize(size = (512,512))
        ])
```

fig. Data preprocessing code snippet

3.4 Model Implementation and Training

For our Model, we have used the U-Net architecture with a small change. Instead of the writing the convolution layer, we have used the partial convolution to aid image inpainting process.

Each layers for our model were implemented with the reference of the Research Paper on Image Inpainting published by Nvidia Coorporation. The implementation details are given below :

Module Name	Filter Size	# Filters/Channels	Stride/Up Factor	BatchNorm	Nonlinearity
PConv1	7×7	64	2	-	ReLU
PConv2	5×5	128	2	Y	ReLU
PConv3	5×5	256	2	Y	ReLU
PConv4	3×3	512	2	Y	ReLU
PConv5	3×3	512	2	Y	ReLU
PConv6	3×3	512	2	Y	ReLU
PConv7	3×3	512	2	Y	ReLU
PConv8	3×3	512	2	Y	ReLU
NearestUpSample1		512	2	-	-
Concat1(w/ PConv7)		512+512		-	-
PConv9	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample2		512	2	-	-
Concat2(w/ PConv6)		512+512		-	-
PConv10	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample3		512	2	-	-
Concat3(w/ PConv5)		512+512		-	-
PConv11	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample4		512	2	-	-
Concat4(w/ PConv4)		512+512		-	-
PConv12	3×3	512	1	Y	LeakyReLU(0.2)
NearestUpSample5		512	2	-	-
Concat5(w/ PConv3)		512+256		-	-
PConv13	3×3	256	1	Y	LeakyReLU(0.2)
NearestUpSample6		256	2	-	-
Concat6(w/ PConv2)		256+128		-	-
PConv14	3×3	128	1	Y	LeakyReLU(0.2)
NearestUpSample7		128	2	-	-
Concat7(w/ PConv1)		128+64		-	-
PConv15	3×3	64	1	Y	LeakyReLU(0.2)
NearestUpSample8		64	2	-	-
Concat8(w/ Input)		64+3		-	-
PConv16	3×3	3	1	-	-

For training the model, we are considering 5 kinds of losses: begin

- Hole Loss - Computes the loss on the masked region
- Valid Loss - computes the loss on unmasked region
- Perceptual Loss - computes the loss of the structure of the image
- Style Loss - computes the loss on the texture of the image
- Total Variational Loss - ensures the smoothness of the pixels in the boundary of mask region and valid region

3.5 Model Testing and Validation

We divided our data set into 3 parts (training, testing and validation). We used the dataset for validation to validate the training of the model and the validation loss is shown in the graph below.

After validation and training we completed we used the testing dataset to test the output of the model. we used new set to data to feed into the model and the output obtained worked in almost every images except in some images with large mask and complex structures.

3.6 GUI Development

We planned on developing the desktop application for our project, so we used PyQT5 python library to create GUI for our project and we used openCV library of python for opening the images. We started by creating the main window that has two buttons "Browse" and "Open" using the function of PyQT5. And we created the functionality to open the dialog box when Browse button was pressed where user can choose the image from device to inpaint. And pressing Open button opens the image for drawing mask. We used "QFileDialog" function of PyQT5 to open the dialog box. In this way we created interactive GUI using PYQt5 and OpenCV.

3.7 GUI and Model Integration

As the GUI of the project and model of the project was created separately we had to integrate them together for completing our project. After the training and testing of the model we downloaded the model and then we integrated the downloaded model and developed GUI in GUImain.py file and it was integrated calling the inpaintFunc() from inpaint.py inside the GUImain.py file when inpaint button was pressed in main window.

Overall, using above methodology we were successful in creating this project with some difficulties in training the model and integrating GUI and model.

4. Experimental Setup

4.1 Dataset Collection:

- Collect a diverse dataset containing images with various degrees of occlusion, lighting conditions, background and object types.
- For training the Model, we need 'Binary Masks' which is a B/W image in which the area covered by the black space represents the portion/patch to be reconstructed. We generated these masks using the OpenCV toolkit and fed it to the learning Model along with the original images for training, testing and validation operations.

4.2 Software Requirements:

Image Processing Libraries:

- PyQt5 and OpenCV: Used for developing a graphical user interface (GUI) enabling manual selection of occluded objects by users and generating masked images.
- Pillow: Employed for handling and processing various image formats, providing functionalities like image opening, manipulation, and saving.

Deep Learning Framework:

- PyTorch, a deep learning framework, is employed to develop and train partial convolutional neural networks (CNNs). These models are specifically designed to identify and eliminate occluded objects within images. PyTorch's flexibility and robustness facilitate the creation of effective algorithms for image reconstruction tasks, enhancing the overall accuracy and efficiency of the system.

Programming Environment:

- Python: Chosen as the primary programming language for scripting and integrating image processing algorithms and machine learning models due to its versatility and extensive libraries support.

4.3 Computational Hardware:

- Local computing using PCs and Cloud-based computing(for training) using Google Colab with high-performance GPUs for accelerated processing, essential for real-time image reprocessing and sufficient RAM to handle large image datasets and support deep learning model training.

4.4 Experimental Workflow:

Data Collection

Gather a diverse dataset containing images with various type of occlusion and various size of occluded objects.

Occluded Object Selection

Design algorithm that allows user to manually draw on image for selection of occluded object and generate the masked image using that image.

Training Deep Learning Models

Train the partial CNN model on the collected dataset to fill the occluded part of the images.

Algorithm Implementation

Combining all the developed algorithm and models and running them for generation of required output.

Validation and Testing

Validate the models and algorithms using a separate set of images not seen during training. And check if the system could handle different type of occlusion or not and if the generated final image is good enough.

4.5 Evaluation Metrics:

4.5.1 Loss

Processing Time:

Evaluate the efficiency of the system by measuring the time taken for image reprocessing, ensuring real-time applicability.

Versatility:

Check whether the system would give correct output for all type of occlusion.

5. System design

This chapter presents the architectural design of the system, outlining its key components and their interactions. The system design aims to fulfill the project requirements and goals effectively. It provides an overview of the high-level structure, including main modules and layers.

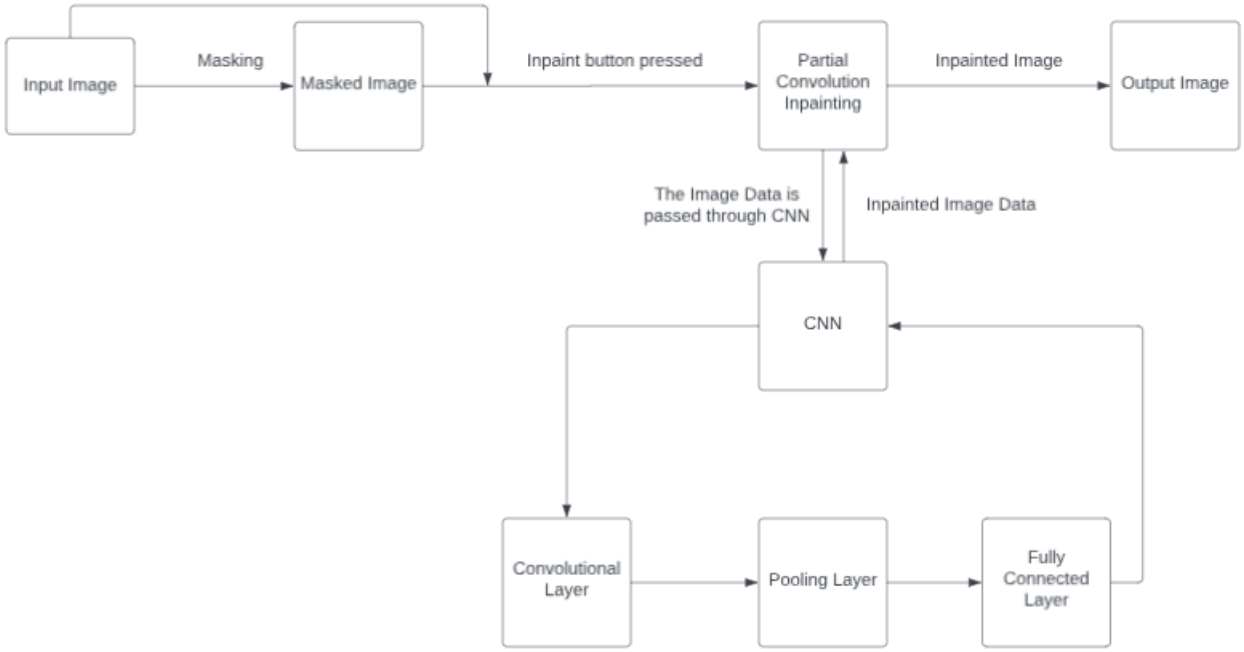

Fig. State Diagram of the System

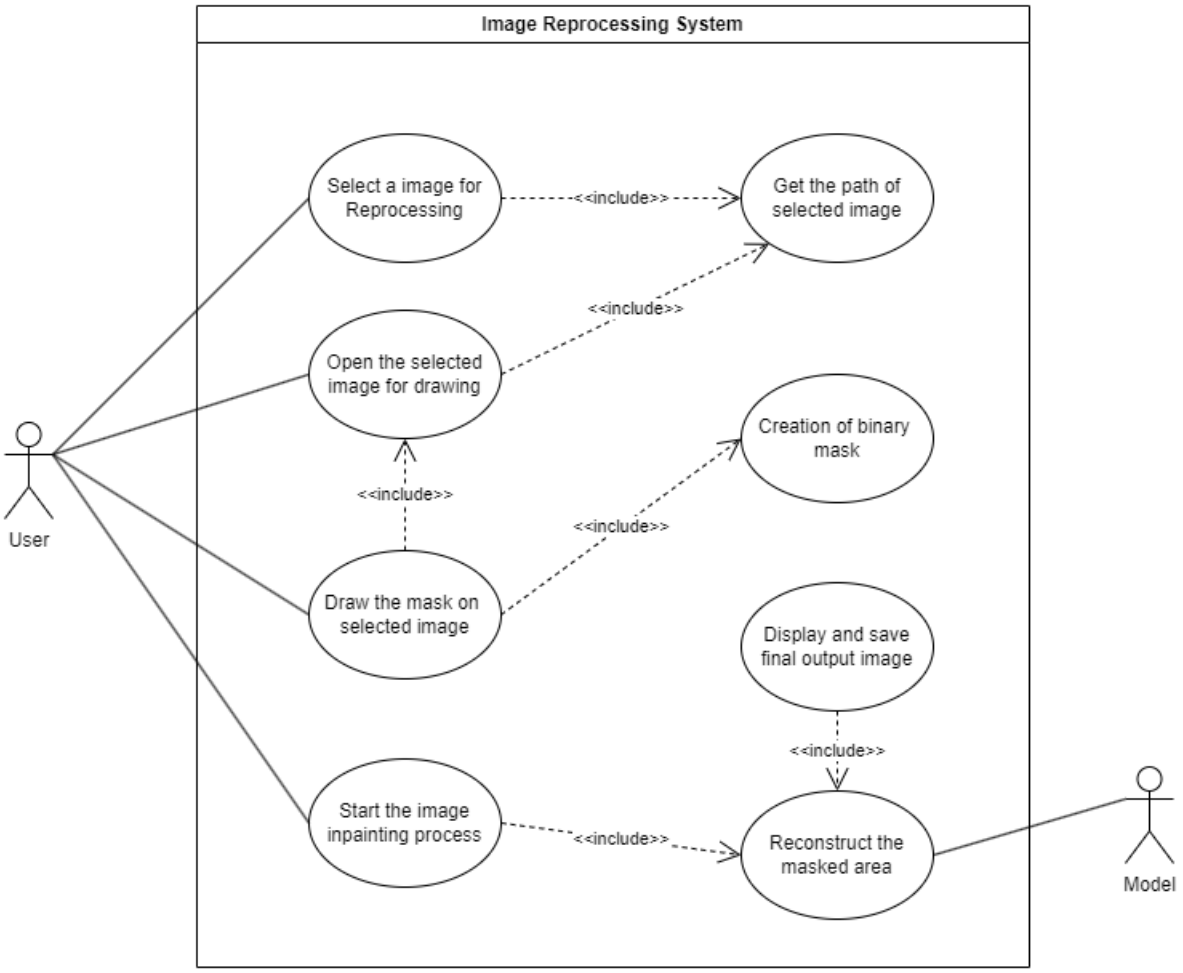

Fig. Use case Diagram of the System

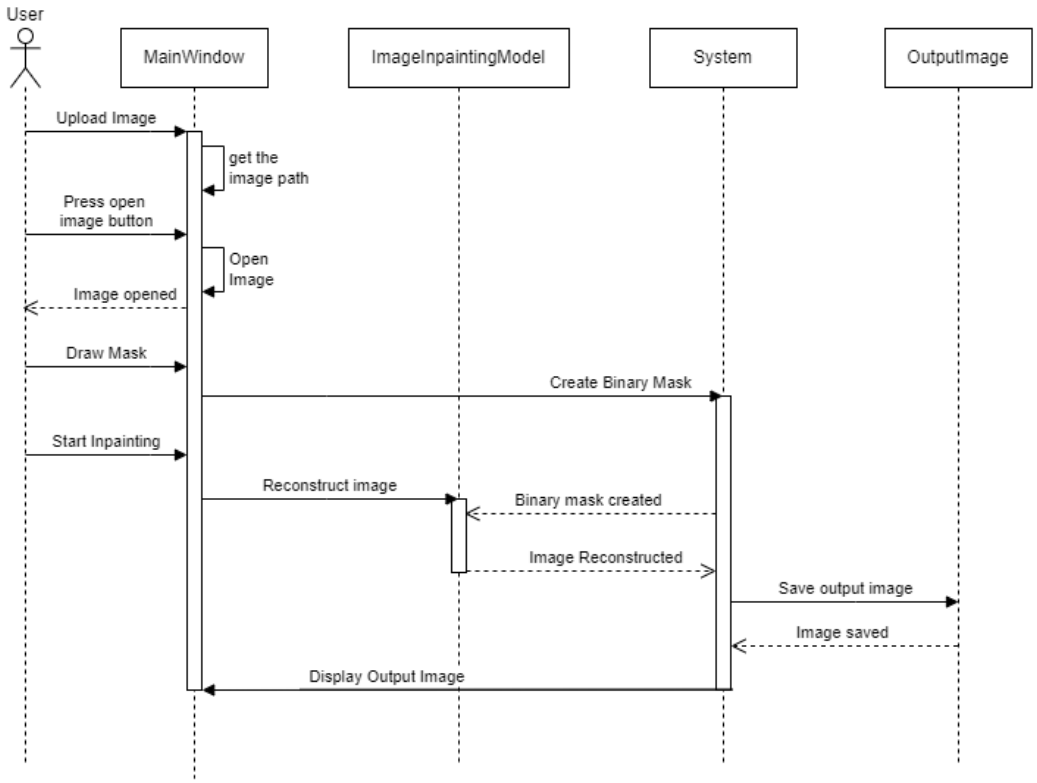

Fig. Sequence Diagram of the System

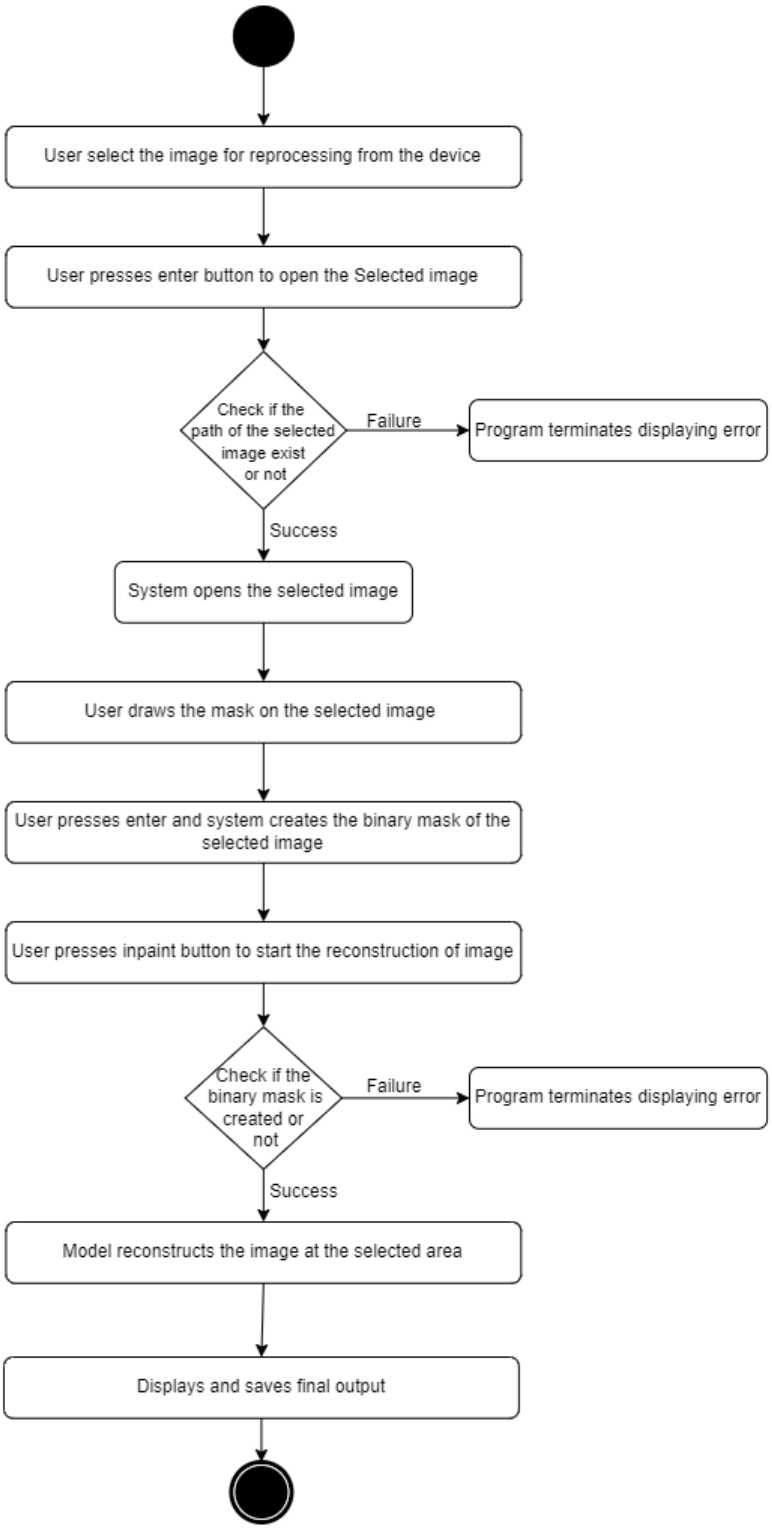

Fig. Activity Diagram of the System

6. Results & Discussion

With the completion of the project, the major objective of the project, inpainting an image was achieved. The software showed promising results for masks with smaller areas .

The final model had a loss of 0.1945.

6.1 Output Results:

Figure Showing the Original and occlusion removed Image

6.2 Loss Analysis

The loss analysis during the training process of our model for image inpainting provides a valuable insights of our model's learning behaiviour. The observed decrease in loss over the tranining epochs indicates that the model is learning to remove the holes from our image.

We trained it for 55 epochs using batch size of 7 with learning rate of 0.0002 for a total of 4200 training datasets and 420 validation dataset (each containing an image and a binary mask).

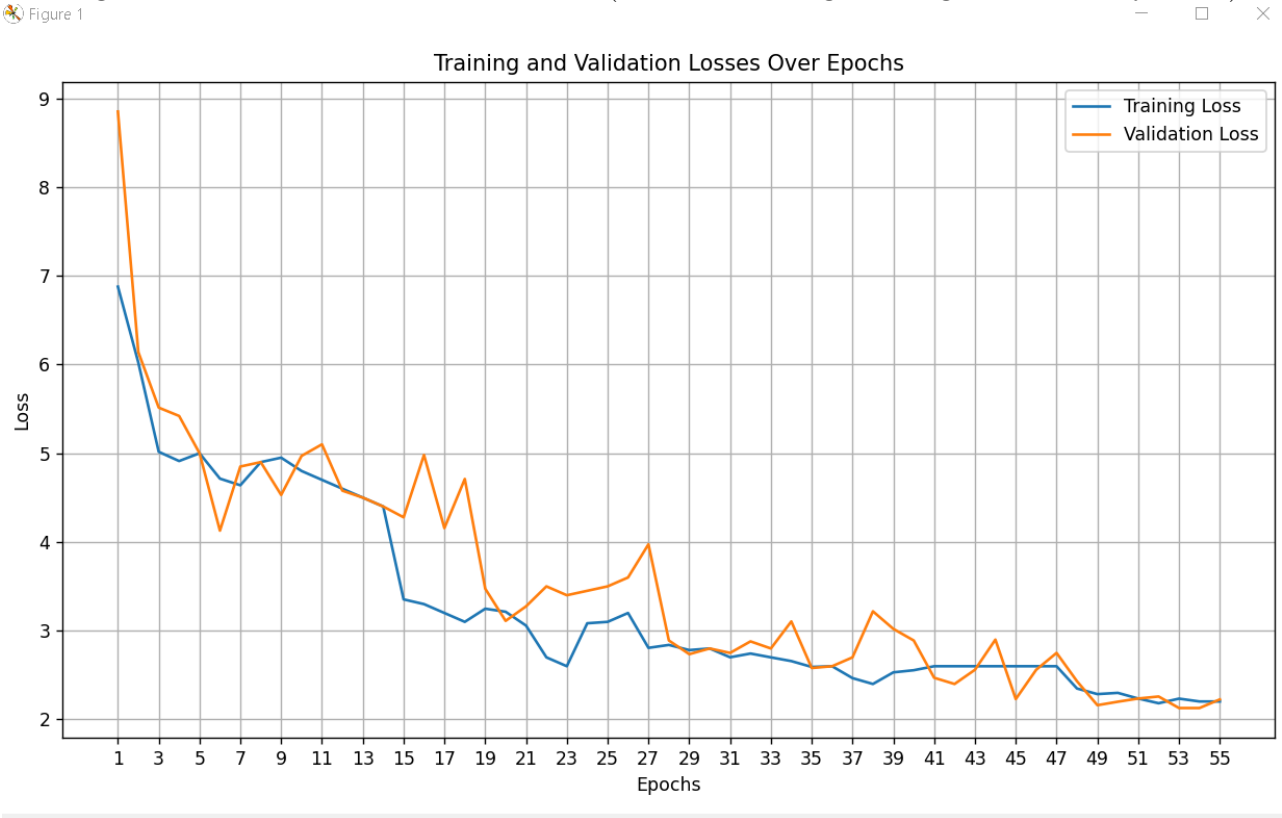

Fig. Loss Curve Diagram of the System Performance

Description:

In the above graph, we can see that during the initial Epochs the loss/error value is really high i.e. around (60-70) Percentage in training datasets and (80-90) Percentage in validation datasets. The error gap between the Training and Validation datasets conveys us that there was over fitting of the learnable parameters during the initial phases. The Losses values are fluctuating and is decrementing over the gradual increment of the number of training epochs. The Problem of over fitting is also settling down and is converged(both training and validation average losses are nearly equal) at the 55th epoch of training. The Loss values will further get reduced when the model is trained over the large and varying datasets which

subsequently increase the accuracy resulting the predicted output of the Model to be much more accurate and precise.

7. Conclusions

In conclusion,the Image inpainting project successfully addressed the challenge of image inpainting using Partial Convolution Technique based on U-Net architecture. Through the development of a user friendly interface and deep learning model, the project was able to accomplish functional requirement for a small masked image with simple background.

Moreover, the project helped us gain insights into various technilogies used in this project such as Machine Learning, Neural Network, PyTorch and Image Processing. Furthermore, it helped us enhance various software engineering skills such as teamwork and project management.

Hence, the project has achieved good results but still there is room for further development and refinement to address limitations and explore additional features. The journey of this project has not only met its technical objectives but has also contributed to the continuous learning and advancement of skills in the field of image processing and deep learning.

8. Limitations and Future enhancement

8.1 Limitations

The model has worked fine in most of the testing dataset with few exceptions. Some of the limitations in this system are:

8.1.1 No Good Output On Large Mask

The developed model faces some problem when it encounters large size mask. It works fine on images with small occluded object (small mask) but when we input the image with large size occluded object (large mask) it does not remove the occluded object successfully and it fails to reconstruct the image with large occluded object.

8.1.2 Complex Images

Based on the tests we conducted on image with complex texture and complex colour patterns, the occluded object was removed but the output obtained was not reconstructed properly satisfactory (Pixels didn't quite match with the surrounding pixels due to complexity of image).

8.2 Future Enchancements

8.2.1 Improvement in GUI

The current GUI is very simple and basic. We can improve the GUI for this project adding some extra features in the main window and improving the visual of the main window by making it look more attractive.

8.2.2 Training on Large Mask

The current model does not work on large mask as we didn't have enough resources to train the model for large mask. So we can improve the model by training the model on large sized mask and testing if it works for large mask or not.

8.2.3 Training on Complex images

The images with complex structure consist different textures and colours and as a result its quite difficult to reconstruct such images. So we can train the model to work on such complex images using complex dataset for training and improve its working on complex images.

Referencing checking here[1] **Referencing checking here**[2]

References

[1] Guilin Liu. et al. Image inpainting for irregular holes using partial convolutions. *NVIDIA Corporation*, page 23, 2018.

[2] Nermin Mohamed Fawzy Salem. A survey on various image inpainting techniques. *Future University*, 2:19, 2021.

Appendices

The Google drive link contains all the code files, and images dataset along with ModelFile.pth which is the progression of the trained model.

https://github.com/BirajKaranjit/Image-Reprocessing-of-Occluded-Object

https://drive.google.com/drive/folders/1Pawp24zh5qY9G8o46$_vM5vXcUR4AoMcr?usp = drive_link$