TECHNICAL UNIVERSITY OF CRETE

DIPLOMA THESIS

Hallucination Detection in Image Inpainting

Author:
Konstantinos ZAFEIRAKIS

Thesis Committee:
Prof. Michalis ZERVAKIS
Prof. Grigorios TSAGKATAKIS
Prof. Thrasyvoulos SPYROPOULOS



A thesis submitted in fulfillment of the requirements
for the diploma of Electrical and Computer Engineer

in the

School of Electrical and Computer Engineering

July 25, 2024

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Hallucination Detection in Image Inpainting

by Konstantinos ZAFEIRAKIS

This thesis addresses the critical issue of hallucinations in deep learning-based image inpainting within remote sensing applications. Remote sensing images are often degraded due to sensor malfunctions or adverse atmospheric conditions. As such, they require inpainting to restore missing information accurately. This restoration is vital for enabling downstream tasks such as classification, detection, and segmentation. Despite the advancements, deep learning models for inpainting face multiple challenges including hallucinations, where the model incorrectly introduces nonexistent elements in the image. This study introduces a novel framework for detecting hallucinations using an image inpainting generator coupled with a two-class discriminator and a class activation mapping (Grad-CAM) model. The experimental setup involves diverse masking techniques and analyzes the inpainting results across different image classes. Our findings reveal significant impacts of mask type and size on hallucination metrics, with rectangular masks generally yielding better results than irregular and random masks. Additionally, each class-specific generator exhibited unique inpainting behaviors, influenced by mask size. The study identifies the in-distribution Dice metric and out-of-distribution prediction value as effective measures for hallucination detection, with the FID metric proving optimal for reconstruction quality.

TECHNICAL UNIVERSITY OF CRETE

Abstract

School of Electrical and Computer Engineering

Electrical and Computer Engineer

Hallucination Detection in Image Inpainting

by Konstantinos ZAFEIRAKIS

Αυτό το άρθρο αντιμετωπίζει το κρίσιμο ζήτημα των παραισθήσεων στην αναπλήρωση εικόνων βασισμένη στη βαθιά μάθηση εντός εφαρμογών τηλεπισκόπησης. Οι εικόνες τηλεπισκόπησης, οι οποίες συχνά υποβαθμίζονται λόγω δυσλειτουργιών των αισθητήρων ή δυσμενών ατμοσφαιρικών συνθηκών, απαιτούν αναπλήρωση για να αποκατασταθεί με ακρίβεια η χαμένη πληροφορία. Αυτή η αποκατάσταση είναι ζωτικής σημασίας για την ενίσχυση επακόλουθων εργασιών όπως η ταξινόμηση, η ανίχνευση και η τμηματοποίηση. Παρά τις προόδους, τα μοντέλα βαθιάς μάθησης για την αναπλήρωση αντιμετωπίζουν προκλήσεις, ιδιαίτερα τις παραισθήσεις, όπου το μοντέλο εισάγει εσφαλμένα αντικείμενα μέσα στην εικόνα. Αυτή η μελέτη εισάγει ένα νέο πλαίσιο για την ανίχνευση παραισθήσεων χρησιμοποιώντας μια γεννήτρια αναπλήρωσης εικόνας σε συνδυασμό με ένα ταξινομητή δύο κλάσεων και ένα μοντέλο Grad-CAM. Το πειραματικό σύστημα περιλαμβάνει διάφορες τεχνικές εφαρμογής μασκών και γίνεται αναλύση για τα αποτελέσματα αναπλήρωσης σε διαφορετικές κατηγορίες εικόνων. Τα ευρήματά μας αποκαλύπτουν σημαντικές επιπτώσεις του τύπου και του μεγέθους της μάσκας στις μετρήσεις παραισθήσεων, με τις ορθογώνιες μάσκες να αποδίδουν γενικά καλύτερα αποτελέσματα από τις ακανόνιστες και τυχαίες μάσκες. Επιπλέον, κάθε γεννήτρια ειδική ως προς μια κατηγορία επέδειξε μοναδικές συμπεριφορές αναπλήρωσης, επηρεασμένες από το μέγεθος της μάσκας. Η μελέτη εντοπίζει τη μετρική Dice εντός κατανομής και την τιμή πρόβλεψης εκτός κατανομής ως αποτελεσματικά μέτρα για την ανίχνευση παραισθήσεων, με τη μετρική FID να αποδεικνύεται βέλτιστη για την ποιότητα ανακατασκευής.

Acknowledgements

It is both an honor and a pleasure to express my gratitude to Dr. Grigorios Tsagkatakis, for welcoming me into SPL and offering invaluable support throughout this thesis. His guidance, insight and patience have been indispensable to the completion of this work. I extend my heartfelt thanks to Dr. Apostolos Dollas for introducing me to FORTH. Finally I gratefully acknowledge the scholarship awarded to me by FORTH.

Contents

Abstract iii

Abstract v

Acknowledgements vii

Contents ix

List of Figures xiii

List of Abbreviations xv

1 Introduction 1

2 Theoretical background 3

2.1 Machine learning . . . 3

2.1.1 Supervised Learning . . . 3

2.1.2 Unsupervised Learning . . . 4

2.1.3 Semi-supervised Learning . . . 4

2.2 Deep learning . . . 4

2.2.1 Artificial Neural Network . . . 4

2.2.2 Artificial Neuron . . . 5

2.2.3 Activation functions . . . 6

2.3 Training Neural Network . . . 8

2.3.1 Batch . . . 8

2.3.2 Loss functions . . . 8

2.3.3 Backpropagation . . . 11

2.3.4 Optimizers . . . 14

2.3.5 Overfitting . . . 15

2.3.6 Underfitting . . . 15

2.4 CNNs - Convolutional Neural Networks . . . 16

2.4.1 Convolutional Layer . . . 17

2.4.2 Pooling Layer . . . 19

2.4.3 Fully Connected Layer . . . 19

2.4.4 Global Average Pooling 20
2.5 Transformers 20

3 Related Work **23**
3.1 Image Inpainting 23
3.2 Remote Sensing Image Classification 24
3.3 Hallucinations 25
3.4 Out of Distribution Detection 25

4 Method **27**
4.1 Overall architecture 27
4.2 Generator 28
4.2.1 Convolutional Head 29
4.2.2 Transformer Body 29
4.2.3 Convolutional Tail 29
4.2.4 Style Manipulation Module 29
4.2.5 Conv-U-Net 29
4.2.6 Generator Loss functions 30
4.3 Classifier 30
4.4 Grad-CAM 31
4.5 Masks 32
4.6 Data and training 33
4.6.1 Generator 33
4.6.2 Discriminator 34
4.7 Experimental Procedure 35
4.7.1 Generator Testing 35
4.7.2 Discriminator Testing 37

5 Experimental Results and Analysis **41**
5.1 Generator Testing 41
5.1.1 CELEBA trained Generator Test on CELEBA input 41
5.1.2 RESISC45 Meadow trained Generator Test on RESISC45 meadow input 44
5.1.3 RESISC45 dense residential trained Generator Test on RESISC45 medium residential input 46
5.2 Generator and Discriminator Testing 49
5.2.1 RESISC45 (not meadow) ground truth to RESISC45 (meadow) trained generator 49
5.2.2 RESISC45 (not dense-residential) ground truth to RESISC45 (dense-residential) trained generator 51

5.2.3 uc_merced (not meadow) ground truth to RESISC45 (meadow) trained generator . . . 53
5.2.4 uc_merced (not dense residential) ground truth to RESISC45 (dense residential) trained generator . . . 54
5.3 Generator and Discriminator Testing varying difficulty scenarios . . 56
5.3.1 Generator trained on RESISC45 meadow input airport 56
5.3.2 Generator trained on RESISC45 meadow input river 58
5.3.3 Generator trained on RESISC45 meadow input forest 59
5.3.4 Generator trained on RESISC45 dense residential input beach 62
5.3.5 Generator trained on RESISC45 dense residential input commercial . . . 64
5.3.6 Generator trained on RESISC45 dense residential input medium residential . . . 65

6 Conclusion **69**

7 Future Work **71**
7.1 Generators . . . 71
7.2 Datasets . . . 71
7.3 Discriminators . . . 71
7.4 Mask types and sizes . . . 72

A Metrics **73**

B Generator Training Instances **77**
B.1 Meadow trained generator . . . 77
B.2 Dense residential trained generator . . . 79

References **81**

List of Figures

1.1 Generator Training and inference 2

2.1 Activation Functions 8
2.2 from left to right underfitting, optimum and overfitting in a classification scenario [23] 16
2.3 Typical architecture of a CNN [27] 17
2.4 Application of a kernel or filter on the input image [24] 18
2.5 Application of max and average pooling [28] 19
2.6 The architecture of the transformer model as designed in the original "Attention is All You Need" paper[32]. 21

4.1 Combined architecture 28
4.2 MAT architecture [60] 28
4.3 Instance of Masks 33
4.4 RESISC45 meadow and dense residential class FID 34
4.5 Generator Training instances for RESISC45 meadow 34
4.6 Generator Training instances for RESISC45 dense residential 34
4.7 Inpainted results, heatmaps, and thresholded heatmaps for a rectangular mask with a generator trained on dense residential and ground truth belonging to an airplane. Each mask is 10% image-wise bigger than the previous 39

5.1 MSE,FID, LPIPS measurements for generator trained on Celeba . . . 41
5.2 Celeba inpainted results 43
5.3 Examples from the meadow class. 44
5.4 MSE,FID, LPIPS measurements for generator trained on RESISC45 Meadow 44
5.5 RESISC45 meadow inpainted results 45
5.6 MSE,FID, LPIPS measurements for generator trained on RESISC45 Medium residential 46
5.7 RESISC45 Medium residential inpainted results 48
5.8 Prediction and dice values for RESISC45 meadow - not meadow . . . 49
5.9 RESISC 45 roundabout inpainted results 50

5.10 Prediction and dice values for RESISC45 dense residential - not dense residential . . . 51
5.11 RESISC 45 beach inpainted results . . . 52
5.12 Prediction and dice values for uc_merced meadow - not meadow . . 53
5.13 uc_merced landuse plane inpainted results . . . 54
5.14 Prediction and dice values for uc_merced dense residential - not dense residential . . . 54
5.15 uc_merced landuse harbor inpainted results . . . 55
5.16 Prediction and dice values for RESISC45 meadow - airport . . . 56
5.17 RESISC45 airport inpainting with meadow trained generator . . . 57
5.18 Prediction and dice values for RESISC45 meadow - river . . . 58
5.19 NPWU RESISC45 river inpainting with meadow trained generator . 59
5.20 Prediction and dice values for RESISC45 meadow - forest . . . 59
5.21 RESISC45 forest inpainting with meadow trained generator . . . 61
5.22 Prediction and dice values for RESISC45 dense residential - beach . . 62
5.23 RESISC45 beach inpainting with dense residential trained generator 63
5.24 Prediction and dice values for RESISC45 dense residential - commercial 64
5.25 RESISC45 commercial inpainting with dense residential trained generator . . . 65
5.26 Prediction and dice values for RESISC45 dense residential - medium residential . . . 65
5.27 RESISC45 medium inpainting with dense residential trained generator 67

A.1 metrics for section 5.3.1 . . . 74
A.2 metrics for section 5.3.2 . . . 74
A.3 metrics for section 5.3.3 . . . 74
A.4 metrics for section 5.3.4 . . . 75
A.5 metrics for section 5.3.5 . . . 75
A.6 metrics for section 5.3.6 . . . 75
A.7 metrics for section 5.2.1 . . . 76
A.8 metrics for section 5.2.3 . . . 76

B.1 Ground Truth images . . . 77
B.2 0 epochs inpainted results . . . 78
B.3 160 epochs inpainted results . . . 78
B.4 Ground Truth images . . . 79
B.5 0 epochs inpainted results . . . 79
B.6 304 epochs inpainted results . . . 80

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
FNN	**F**eedforward **N**eural **N**etwork
RELU	**R**ectified **L**inear **U**nit
ML	**M**achine **L**earning
RL	**R**einforcement **L**earning
CNN	**C**onvolutional **N**eural **N**etwork
ID	**I**n **D**istribution
OOD	**O**ut **O**f **D**istribution
GAN	**G**enerative **A**dversarial **N**etwork
SGD	**S**tochastic **G**radient **D**escent
NLP	**N**atural **L**anguage **P**rocessing
CAM	**C**lass **A**ctivation **M**ap
OCC	**O**ne **C**lass **C**lassification
MSE	**M**ean **S**quare **E**rror
LPIPS	**L**earned **P**erceptual **I**mage **P**atch **S**imilarity
FID	**F**réchet **I**nception **D**istance
MAE	**M**ean **A**bsolute **E**rror

Chapter 1

Introduction

Image inpainting refers to the process where damaged, deteriorated, or missing parts of an image are filled in to present a complete image. Modern use of inpainting can be traced back to the 18th century used to restore artworks in Venice. More recently with the advent of deep learning image inpainting can be done in digital images with increased accuracy [1]

In remote sensing, deep learning based inpainting methods are the current gold standard and are preferred over more traditional image inpainting methods [2]. Deep Learning model are able to handle both the high resolution of remote sensing data and the geographical variations characterizing these images, which render conventional inpainting methods less effective [3]. This entails the requirement of models with high complexity to sufficiently capture the spectral, spatial, and textural nuances within an image emerging from its high spatial variability. Image inpainting in remote sensing has a lot of uses since sensor malfunctions and poor atmospheric conditions such as cloud obstruction lead to the loss of large regions [4]. Image inpainting helps in filling these missing regions with semantically coherent content which can improve the performance of downstream tasks such as classification, detection, and segmentation with real-life uses like crop yield modeling [5] Civil Engineering [6] mineral exploration [7]

The deep learning inpainting procedure however has its own challenges hallucinations being one of them. The concept of hallucinations pertains to a perceptual phenomenon in which a machine learning model commits a classification error by assigning an object depicted within an image to a class that deviates from its genuine categorization. This misattribution leads to the model categorizing the object as a constituent of an alternative class, diverging from its authentic classification. Such misclassifications can limit the value of image inpainting of remote sensing images.

In this work, we perform a comprehensive investigation into the phenomenon of hallucinations which is conducted through the utilization of an architecture consisting of an image inpainting generator in conjunction with a two-class discriminator and a class activation mapping (Grad-Cam) model. In the proposed model, the generator is trained on a different dataset than the input image inpainting a different class of image within the original ground truth as seen for example in fig 1.1. With this architecture, we evaluate the likelihood of an image belonging to each respective class in addition to class activation map analysis. Our methodology involves applying diverse masking techniques to images of distinct classes, followed by measurement and analysis of the ensuing results.

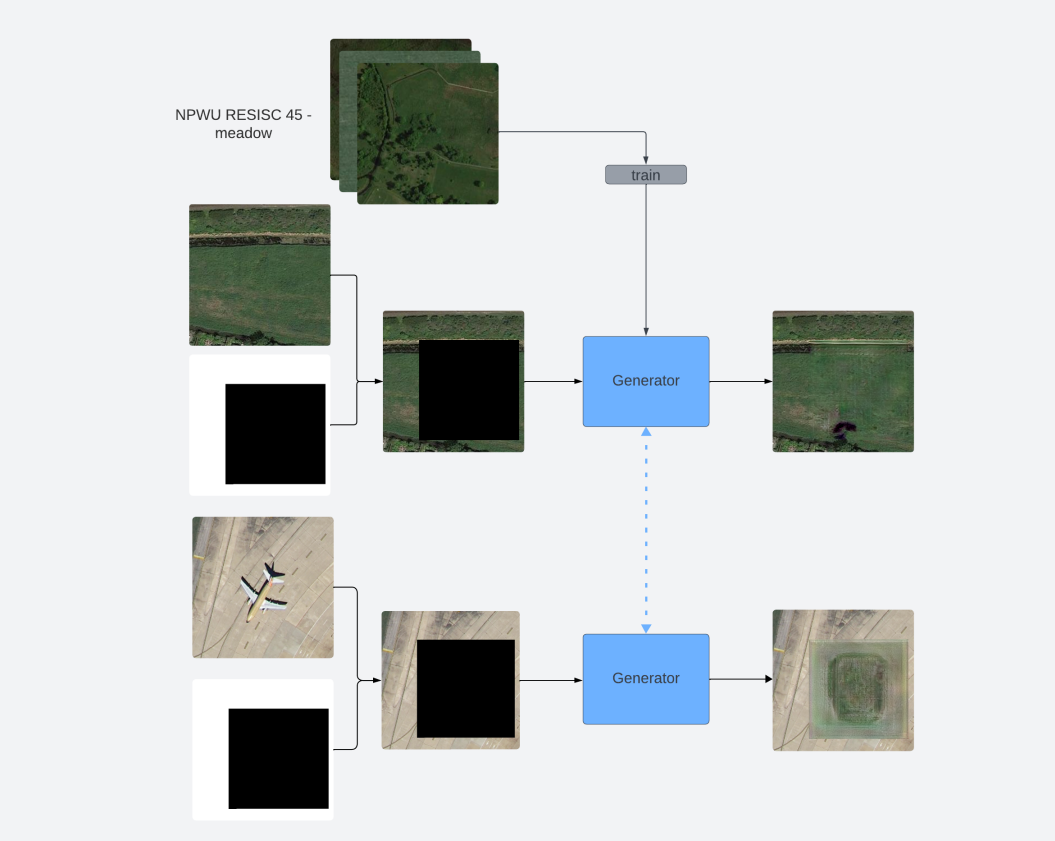


FIGURE 1.1: Generator Training and inference

To our knowledge, no research on hallucination detection in a remote sensing environment has been reported in the literature. Therefore, the main contributions of this work include 1) the development of a framework for hallucination detection in remote sensing, 2) a thorough investigation of the problem of hallucinations, and 3) an extensive experimental evaluation in a remote sensing dataset.

Chapter 2

Theoretical background

2.1 Machine learning

Machine learning or ML, a subfield of artificial intelligence, is a computational approach that enables computers to make predictions without being explicitly programmed to do so in specific tasks. Machine learning makes use of algorithms and statistical models to learn from data identifying patterns that enable systems to adapt and improve their performance.

Uses for Machine learning have seen a sharp rise in recent years largely due to the emergence of deep learning which involves training deep neural networks on large datasets to achieve state-of-the-art results. Machine learning is applied in many fields such as natural language processing, computer vision, recommendation systems, healthcare, finance, and more to predict, identify and analyze data [8].

Machine learning can be divided into three fundamental paradigms, Supervised, Unsupervised, and Semi-Supervised Learning, each characterized by different methodologies and applications.

2.1.1 Supervised Learning

In supervised learning, the model is trained on a labeled dataset where each input data example is associated with a corresponding label, typically based on human annotation. While supervised learning models tend to be more accurate than unsupervised learning models, for large datasets, the cost of leveraging a supervised learning model is prohibitive. Supervised learning models are often used in classification (assigning data to a class) and regression (predicting a continuous value) problems such as image recognition or stock price prediction.

2.1.2 Unsupervised Learning

Unsupervised learning is a machine learning paradigm where the model is trained on an unlabeled dataset with an objective of discovering patterns and relationships within the data without specific guidance or target labels. Unsupervised learning is the preferred method when a dataset is vast. The type of problems it is used for are clustering (grouping similar data points), anomaly detection (identification of unusual data points), and feature extraction (reduction of the dimensionality of data without important information loss) problems such as principal component analysis (PCA), customer segmentation and recommendation engines.

2.1.3 Semi-supervised Learning

Semi-supervised learning is a machine learning paradigm and is the middle ground between supervised and unsupervised learning. In semi-supervised learning, a model is trained on a dataset that contains both labeled and unlabeled data. Semi-supervised learning combines the benefits of both labeled data which provides clear examples for the model to learn from and unlabeled data which is often more abundant and can help the model generalize better. Semi-supervised learning models are often used in in cases where the dataset is far too large to be labeled fully therefore by leveraging Semi-supervised Learning the dataset is partially labeled and the advantages of both supervised and unsupervised learning are utilized.

2.2 Deep learning

Deep learning is a sub field of machine learning which focuses on training artificial neural networks (ANN) which are networks inspired by the brain. Much like conventional machine learning, deep learning encompasses various learning paradigms including supervised, unsupervised, and semi-supervised learning.

Deep learning has seen a significant rise in popularity due to rapid advancements in the field which have been significantly catalyzed by the proliferation of cost-effective computing power. These advancements have enabled deep learning to be applied to a diverse array of domains.

2.2.1 Artificial Neural Network

Artificial neural networks or neural networks are computational models inspired by the function and structure of the brain. Neural networks are organized into multiple layers with each layer comprising numerous interconnected artificial neurons, the

layers are organized into an input layer which receives the information to be processed, multiple hidden layers which perform the necessary computations and an output layer which provides the result. The artificial neurons collectively form the fundamental building blocks of the network and play a pivotal role in the function of the neural network.

Different architectures of neural networks have been developed to address specific tasks such architectures are Feedforward Neural Networks (FNNs) which are the simplest form of neural networks and are used for various tasks such as regression and classification. Convolutional Neural Networks (CNNs) are tailored for computer vision tasks as they efficiently handle grid-like data such as images. Recurrent Neural Networks (RNNs) are designed for sequential data since they can capture temporal dependencies making them ideal for tasks like natural language processing.

2.2.2 Artificial Neuron

An artificial neuron [9] also known as a node or perceptron is the fundamental unit within an artificial neural network. It is designed to emulate the behavior of a biological neuron in the brain in a simplified manner.

In a neural network, neurons are organized into layers, and each neuron can be formally defined as a mathematical function. This function takes as input the outputs of neurons from the preceding layer, and it performs the following operations:

1. **Weighted Summation**: The neuron computes the weighted sum of its inputs. Each input is multiplied by a corresponding weight, and these weighted inputs are summed together.
2. **Bias Addition**: To this weighted sum, a bias term is added. The bias term allows for fine-tuning the neuron's activation threshold and can shift the neuron's response.
3. **Activation Function Application**: The result of the weighted sum plus the bias is then passed through an activation function or a transfer function. The activation function makes the neuron's output non-linear and transforms the weighted sum into the final output of the neuron. There are numerous activation functions commonly used such as sigmoid, ReLU and more.

In mathematical terms the output of the k-th neuron can be represented as:

$$y_k = \phi\left(\sum_{i=0}^{m}(w_{ki} \cdot x_i) + b\right) \tag{2.1}$$

Where:

- y_k is the output of the k-th neuron
- ϕ represents the activation function
- w_{ki} is the weight associated with the connection between the i-th input neuron and the k-th neuron.
- x_i represents the output of the i-th neuron from a previous layer.
- b represents the bias of the k-th neuron

2.2.3 Activation functions

Activation functions [10] [11] are an integral part to the output of each neuron most activation functions used are non-linear since it enables neural networks to learn complex non-linear patterns in data. Without non-linear activation functions a deep neural network would be a linear model which is severely limited. Activation functions usually have activation ranges between 0 and 1 or -1 and 1 some of the more important activation functions are the sigmoid, tanh, ReLU and leaky ReLU functions

Sigmoid activation function The sigmoid activation function is a classic activation function used for logistic regression and is characterized by the following formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

The sigmoid function produces outputs between 0 and 1 and has a smooth continuous form one of its problems is that it is not zero-centered meaning that its output is always positive which can lead to slower convergence during training. The most important drawback however is that the sigmoid function suffers from the vanishing gradient problem meaning that when the function values are too high or too low the derivative of the functions becomes very small which causes vanishing gradients and hinders learning. The vanishing gradient is one of the main reasons the sigmoid is not typically used in deep learning networks. Graph of the sigmoid function can be seen in the table **??**

Tanh activation function The tanh or Hyperbolic Tangent Activation Function is another popular activation function described by the following formula:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$

The tanh function produces outputs between -1 and 1, and similar to the sigmoid function the tanh function is smooth and continuous. Although the tanh function still suffers from the vanishing gradient problem due to its bigger range it fairs better than the sigmoid. A major advantage the tanh exhibits over the sigmoid is that it is zero-centered meaning it can map negative values which can be advantageous in many problems where strong positive and negative correlations are present. Graph of the tanh function can be seen in the table **??**

ReLU activation function The ReLU or Rectified Linear Unit activation function is a piecewise-linear function and is one of the more popular activation functions in deep learning. The formula for the ReLU function is:

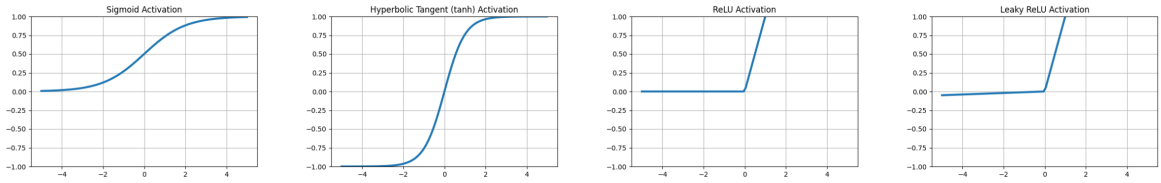
$$ReLU(x) = \begin{cases} x, & x \geq 0. \\ 0, & x < 0. \end{cases} \tag{2.4}$$

Despite its piecewise-linear appearance, the ReLU function is non-linear due to the abrupt change in behavior at the origin point. The ReLU has significant advantages over the sigmoid and tanh, one is that it is computationally efficient since no exponents are calculated. Moreover, for positive values, there is no gradient vanishing problem since for every positive value the gradient is one. The ReLU however is not zero-centered as negative values are clipped which introduces the dead ReLU problem where a neuron consistently outputs zero for all inputs. This can happen when a large gradient flows through a ReLU neuron causing the weights to be updated in a way that the neuron never activates. Graph of the ReLU function can be seen in the table **??**

leaky ReLU activation function The leaky ReLU activation function is specifically designed to compensate for the dying ReLU problem the mathematical formula of the leaky ReLU is:

$$LReLU(x) = \begin{cases} x, & x \geq 0. \\ \alpha \cdot x, & x < 0. \end{cases} \tag{2.5}$$

Leaky ReLU is a piecewise-linear activation function which despite containing linear segments is considered a non-linear activation function because it introduces a degree of non-linearity by allowing a small non-zero gradient α for negative inputs, the value of α is usually 0.01. Due to the addition of α leaky ReLU is capable of producing as an output value any real number. The main advantage of leaky ReLU is that it addresses the dying ReLU problem theoretically leaky ReLU should have all the advantages of ReLU in actual operation however it seems that Leaky ReLU is not always better than ReLU. Graph of the Leaky ReLU can be seen in the table **??**



(a) Sigmoid (b) Tanh (c) ReLU (d) Leaky ReLU

FIGURE 2.1: Activation Functions

2.3 Training Neural Network

Training a neural network involves adjusting all of its parameters i.e. its weights and biases, which are initially assigned random values. In supervised learning, the first step in parameter adjustment is to define a loss function. The loss function quantifies how much the network's output diverges from the ground truth. A larger divergence lead to a higher value of the loss function.

To iteratively optimize the network's parameters the backpropagation algorithm is typically used. Backpropagation calculates the gradient of the value of the loss function by doing a forward pass to get the loss value and a backward pass to calculate the gradient. Once the gradient of the loss is calculated, it is used by an optimization algorithm to update the values of the network's parameters.

The process of training a neural network is iterative and these steps are repeated for every batch of the dataset and for multiple epochs (instances of the dataset) until the model's performance converges to a satisfactory level.

2.3.1 Batch

A batch is a discrete subset of the dataset. The reason for using batches in every iteration of the training process stems from the computational impracticality of computing the gradient for the entire dataset in a single step. Therefore the dataset is shuffled and then divided into smaller manageable batches. Then the gradient is computed upon these batches saving computational resources. However the gradient computed in this manner is not the true gradient as it does not encompass the entire dataset in a single iteration but it constitutes a highly accurate approximation of it.

2.3.2 Loss functions

Loss functions or cost functions [12] are a crucial component of machine learning algorithms. They are used to measure the disparity between the predicted values

and the actual target values or ground truth in a supervised learning task. The primary goal of a machine learning model is to minimize the loss function as doing so results in increased accuracy and better predictions. In supervised learning loss functions are categorized into two primary groups each corresponding to a type of neural network these are regression and classification loss functions.

- **Regression Loss Functions**: used in regression neural networks which are neural networks used for solving regression type problems meaning that given an input value the model predicts a corresponding output value
- **Classification Loss Functions**: used in classification neural networks which are neural networks used for solving classification type problems meaning given an input the neural network produces a vector of probabilities each corresponding to pre-set categories

Regression Loss Functions

Mean Squared Error (MSE) Mean Squared Error is a popular loss function used to measure the average squared difference between the actual values and the predicted values of a dataset by squaring the difference of them and dividing them by the number of observations the MSE in mathematical form can be seen below

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.6}$$

where:

- n is the number of observations
- y_i is the ith observed value
- $\hat{y}_i$ is the corresponding predicted value

Some of the benefits of the Mean Squared Error function is that since the difference is squared MSE measures only the magnitude of the deviation from the actual value making the sign of the prediction irrelevant. Also MSE is a convex function with a well-defined global minimum which allows for easier utilization of gradient decent optimization. However the MSE is susceptible to the influence of outliers since the difference is squared. In cases where a single predicted value deviates substantially from the actual value it can disproportionately increase the overall loss.

Mean Absolute Error (MAE) Mean Absolute Error is another popular error function that finds the average of the absolute differences between the target and the

predicted outputs the MAE in mathematical form can be seen below:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.7}$$

where:

- n is the number of observations
- y_i is the ith observed value
- $\hat{y}_i$ is the corresponding predicted value

Since the Mean Squared Error loss function is susceptible to outlier predictions due to the distance being squared the Mean Absolute Error function which is not as susceptible is often favored as an alternative. Notable instances occur when the training dataset includes a prevalence of outlier data points. However as the average absolute distance between predicted and actual values approaches zero gradient descent optimization will not work due to the fact that the derivative of the MAE function at zero is undefined.

Classification Loss Function

Binary Cross-Entropy Binary Cross-Entropy is a loss function used in binary classification models where the model is tasked with categorizing an input data point into one of two predetermined categories. In binary classification the output values y_i are either zero or one to quantify the loss between these values and the predicted probability the loss function must compare the actual value with the probability that the input aligns with the category or not. The Classification Loss Function in mathematical form can be seen below:

$$\text{BE Loss} = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot log\,(p_i) + (1 - y_i) \cdot log\,(1 - p_i) \tag{2.8}$$

Where:

- n is the number of observations
- p_i is the probability of one
- $1 - p_i$ is the probability of zero

Categorical Cross-Entropy Categorical Cross-Entropy [13] is a loss function used in multi-class classification tasks the goal is to categorize input data points into one of several classes

$$CELoss = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{C} y_{ij}\log\left(\hat{y}_{ij}\right) \tag{2.9}$$

where:

- N is the number of observations
- C is the number of classes
- y_{ij} is the true output for the ith sample and the jth class
- $\hat{y_{ij}}$ is the predicted output for the ith sample and the jth class

2.3.3 Backpropagation

The backpropagation algorithm [14] [15] [16] is used to compute the gradient of the loss function for each element of a given batch with respect to the parameters. Subsequently the median gradient is computed for all the elements in that batch thus producing the gradient of the batch. The process through which backpropagation calculates this batch gradient has both a forward and a backward pass through the neural network.

- **Forward Pass**: In this step the input data is fed through the neural network. The network computes its predictions and the output is compared to the ground truth. The loss function evaluates the disparity between the predicted and target values.
- **Backward Pass**: After the forward pass the gradient of the loss function with respect to each parameter is calculated. The chain rule of calculus is used to propagate the error backward through the layers of the network. This process enables the network to assign responsibility for the observed errors to each specific parameter

The subsequent analysis will analyze backpropagation using the chain rule and matrices.
Denotations:

- w^l_{jk} is the weight for the connection of the kth neuron belonging in the $(l-1)^{th}$ layer to the jth neuron belonging in the lth layer.
- b^l_j is the bias of the jth neuron belonging in the lth layer.
- a^l_j is the activation of the jth neuron belonging in the lth layer.

As explained before the activation of the jth neuron in the lth layer can be interpreted as the activation function of the sum of all the activations of the previous layers

multiplied by their respective weights with the addition of specific neuron's bias.

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \text{ therefore } a_j^l = \sigma(z_j^l) \tag{2.10}$$

The equation above can be generalized layer wise in a matrix form by defining an activation matrix a^l with each element corresponding to the activation of a neuron in the lth layer similarly a^{l-1} and b^l matrices are created as well as w^l which is a layer matrix with each element corresponding to the weights connecting to the l layer of neurons .

$$z^l = w^l a^{l-1} + b^l \text{ therefore } a^l = \sigma(z^l) \tag{2.11}$$

Since the Backpropagation algorithm calculates the gradient of the loss with respect to each parameter every partial derivatives $\frac{\partial C}{\partial w_{jk}^l}$ and $\frac{\partial C}{\partial b_j^l}$ must be calculated for each weight and bias. A cost function C is introduced that can be written as a function of the outputs of the final layer of the network. Also an intermediate quantity δ_j^l called error of the jth neuron belonging to the lth layer is introduced to simplify the calculations specifically four equations using the error will be used. The δ_j^l error can be defined as

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \tag{2.12}$$

Equation of the output layer error Through application of the chain rule we get

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} \rightarrow \delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{2.13}$$

And since $\alpha_j^L = \sigma(z_j^L)$ the equation becomes

$$\delta_j^L = \frac{\partial C}{\partial z_j^L}\sigma'(z_j^L) \tag{2.14}$$

Written in a matrix format

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{2.15}$$

Where $\odot$ is the hadamard product or element wise product of two vectors and $\nabla_a C$ is a vector whose elements are partial derivatives $\frac{\partial C}{\partial a_j^L}$.

Equation of the error in terms of the error of the next layer Similarly through application of the chain rule we get

$$\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l) \tag{2.16}$$

Where $(w^{l+1})^T$ is the transpose matrix of w^{l+1}. This equation allows knowing the error and the weights of the $(l+1)^{th}$ layer to calculate the error of the lth layer by multiplying the transposed weights and error and taking the Hadamard product $\odot\sigma'(z^l)$. This way the error can be calculated backwards from a layer whose parameters and error are known.

Equation for change of cost in terms of any bias Through application of the chain rule we get

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \tag{2.17}$$

Written in matrix form

$$\frac{\partial C}{\partial b} = \delta \tag{2.18}$$

This equation allows the computation of the partial derivative of the cost with respect to bias of a neuron as a function of the error of the same neuron which is calculated using equations 2.15 and 2.16

Equation for change of cost in terms of any weight Through application of the chain rule we get

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{2.19}$$

This equation allows the computation of the partial derivative of the cost with respect to the weight of a neuron as a function of the error of the same neuron and the neuron of the same position of the previous layer. The error can be calculated in the backward pass using equation 2.15 and 2.16 and the activation is calculated in the forward pass.
Using the above equations allows us to calculate the partial derivatives of the cost with respect to weights and biases as shown below

- **Forward Pass**: for each layer l=2,3...L compute $z^l = w^l a^{l-1} + b^l$ as well as $a^l = \sigma(z')$
- **Final Layer error**: the error of the final layer is computed $\delta^L = \nabla_a C \odot \sigma'(z^L)$
- **Backward Pass**: for each layer l= L,L-1...2 the error of each layer is computed $\delta^l = ((w^{l+1})^T \delta^{l+1} \odot \sigma'(z^l)$
- **Gradients**: Compute the gradient of cost with respect to the network parameters as $\frac{\partial C}{\partial b^l_j} = \delta^l_j$ and $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j$

Leveraging the above process the gradient is calculated and subsequently utilized as an input to the optimizer facilitating the adjustment of parameters.

2.3.4 Optimizers

Optimizers [17] [18] [19] are algorithms used to adjust the parameters of a neural network in order to minimize the loss function.

Gradient Descent Gradient descent is an iterative optimization algorithm designed for the minimization of a convex function. It iteratively adjusts its parameters in order to approach the local minimum of a specified function. This process involves the gradual reduction of a loss function by navigating in the opposite direction to the steepest ascent within the function's landscape. The gradient descent relies on the derivatives of the loss function to calculate the locations of minima . Gradient descent has hyperparameters such as the learning rate that need to be carefully tuned to ensure efficient and stable optimization. Setting the learning rate too high can lead to overshooting the minimum while setting it too low can cause slow convergence. .The mathematical form of gradient descent can be seen below

$$W_{new} = W_{old} - a \cdot \frac{\partial(Loss)}{\partial W_{old}} \tag{2.20}$$

Where:

- W_{new} is the new weight or bias
- W_{old} is the old weight or bias
- a is the learning rate
- $Loss$ is the loss of the loss function

However since weights are updated subsequent to the computation of gradients of the entire dataset. In cases where the algorithm is applied to big datasets it will take a lot of time to converge to the minimum. Also to compute the gradients of the entire dataset is a very memory intensive process.

Stochastic Gradient Descent (SGD) is a variation of Gradient descent that updates the parameters of the network after each data point instead of the whole dataset. Meaning that it can be applied in large scale datasets and is memory efficient however it has variance and it's frequent updates are computationally expensive.

Mini-Batch Gradient Descent splits the training dataset into small batches and performs an update for each of those batches. Mini batch is a balanced algorithm that requires a medium amount of memory and updates the parameters frequently and with less variance. However it has an increased chance to get trapped at local minima and the convergence speed is heavily reliant on the learning rate.

2.3.5 Overfitting

Overfitting [20] [21] occurs when a model fits exactly against its training data. In the framework of a machine learning algorithm, a model is trained with a representative data set. But if the model gets extended training on this data set or the data set is too complex, it can begin to absorb "noise" or irrelevant information in the data, the model then begins to memorize this noise and leaves itself open to overfitting which is too close to the training data.

Overfitting can be prevented by diversifying and scaling the training data set or using some other strategies like the following

- **Early stopping**: Early stopping is a technique that intervenes in the training phase to prevent a machine learning model from learning the noise in the data. However this approach risks halting the training process too soon which leads to the opposite problem of underfitting.
- **Regularization**: Regularization is a collection of training/optimization techniques that seek to reduce overfitting by reducing the complexity of the model through reducing the number of features. Regularization applies a penalty to the input parameters with the larger coefficients, which effectively limits the amount of variance in the model. Varieties of regularization methods exist like L1 and Lasso regularization as well as dropout.
- **Train with more data**: Expanding the training set to include more data can increase the accuracy of the model by providing more samples to identify significant relationships between input and output variables though this approach is more powerful when the added data is relevant. Otherwise, constantly increasing the data set can introduce unnecessary complexity, which can lead to overfitting problems.
- **Data augmentation**: Data augmentation changes the sample data slightly every time the model processes it. This is done by changing the input data in small ways. Using data augmentation in moderation way adds uniqueness to the training data, which helps prevent the model from becoming overly familiar with the data characteristics. Examples of these transformations are flipping and rotation to input images.

2.3.6 Underfitting

Underfitting [22] occurs when a model doesn't accurately capture the relationship between input and output variables. This leads to a high error rate on both the training set and unseen data. The reason often is that a model is being overly simplistic,

which can be a result of inadequate training time, insufficient input features or excessive regularization. Similar to overfitting underfitting prevents the model from grasping the dominant trends in the data, resulting in training errors and reduced model performance.

Underfitting can mainly be countered by increasing the complexity of the model below are some methods that can help reduce underfitting

- **Regularization Decrease**: Regularization as mentioned before is a technique used to curb the variance of a model by imposing penalties on input parameters with larger coefficients. However, when the data features become overly uniform, the model struggles to identify the prevailing trend, resulting in underfitting. To combat this, reducing the level of regularization increases complexity and variation of the model, allowing it to detect important patterns in the data.
- **Training Duration Increase**: Ending the training process prematurely can cause the model to underfit. By extending the training duration, which allows the model to capture more intricate relationships within the data underfitting can be countered. There must be struck a balance for the training time since too little can cause underfitting while too much can cause overfitting.
- **Feature selection**: With any model, specific features are used to determine a given outcome. When the set of predictive features is insufficient additional features could be added. For example in the context of a neural network more hidden neuron layers can be added to increase the complexity of the model and enable it to find more patterns in the data.

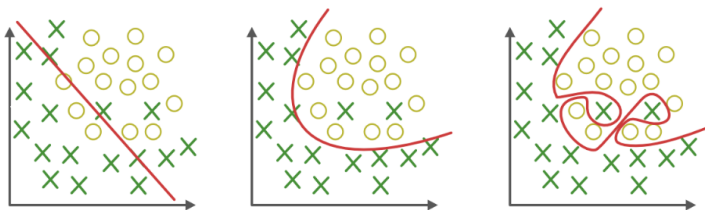


FIGURE 2.2: from left to right underfitting, optimum and overfitting in a classification scenario [23]

2.4 CNNs - Convolutional Neural Networks

A Convolutional Neural Network[24] [25] [26], also known as CNN or ConvNet, is a class of neural network optimized in extracting features from data that has a grid-like topology, such as an images via kernel or filters optimization. CNNs have three main layer types Convolutional layers, Pooling layers and Fully connected layers.

The initial component of a convolutional network is the convolutional layer. Subsequently additional convolutional or pooling layers can be added while the final layer is comprised of a fully connected layer. With each passing layer, the CNN identifies more and more complex patters until it finally identifies the intended object.

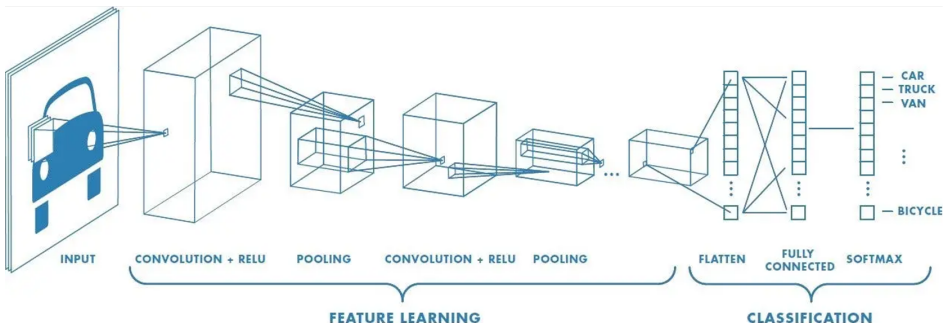


FIGURE 2.3: Typical architecture of a CNN [27]

2.4.1 Convolutional Layer

The Convolution Operation plays a crucial role in extracting high-level features from input images in Convolutional Neural Networks. Usually in a CNN multiple Convlayers are employed to enhance feature extraction. Conventionally, the first ConvLayer focuses on capturing low-level features like edges, colors and gradient orientation. As more layers are added the architecture evolves to recognize high-level features resulting in a network with a comprehensive understanding of images akin to how humans perceive them.

The input of the first ConvLayer is an image which has three dimensions height, width, and depth (RGB values) a kernel or filter is applied which will move across the image, checking if the feature is present. The kernel is a two dimensional matrix of weights and has varying sizes which also determines the size of the receptive field. When the kernel is applied to an area of the input image the hadamard product is calculated between the input pixels and the filter. This product is then fed into an output array or activation map. During the forward pass, the kernel slides across the height and width of the image filling out all the positions of the activation map, the step by which the kernel moves across the image is called Stride. For the first Convolutional layer since the input image is three dimensional with three elements for depths the image is split to tree two dimensional matrices. The kernel is applied to each one and their output are summed to form each element of the activation map.

There are instances where the dimensionality of the input matrix does not correspond to that of the kernel in which cases an appropriate padding must be applied

- **Valid padding**: Valid padding also known as no padding. In this case, the last convolution is dropped if dimensions do not align.
- **Same padding**: Same padding ensures that the output layer has the same size as the input layer
- **Full padding**: Full padding increases the size of the output by adding zeros to the border of the input.

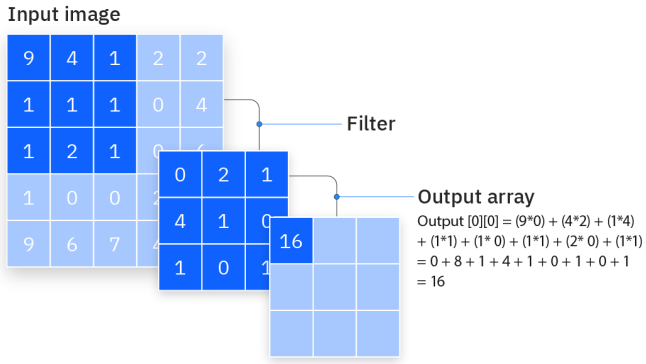


FIGURE 2.4: Application of a kernel or filter on the input image [24]

The Convolution operations leverages three important topics: sparse interaction, parameter sharing, and equivariant representation.

- **Sparse Interaction**: CNNs through having the dimensionality of the kernel smaller than that of the input achieve spare interaction meaning for example that an image consisting of thousands or millions of pixels when processed using the kernel meaningful information can be detected that is of tens or hundreds of pixels. This means that fewer parameters can be stored which reduces the memory requirement of the model but also improves its efficiency.
- **Parameter Sharing**: During the process of creating the activation map, the neural weights of the kernel remain fixed as it moves across the image. In a traditional neural network each element of the weight matrix is used once and then never revisited but a convolution network has shared parameters since the same kernel is used to scan the entirety input image, though the weights change during the training process.
- **Equivariant Representation**: Due to parameter sharing, the layers of convolution neural network have the property of equivariance meaning that a change of the input in a some way will result in the output being changed also in the same way.

2.4.2 Pooling Layer

Pooling layer's primary function is to reduce the spatial dimensions of the input thus minimizing the number of parameters involved and simplifying the information within a CNN. Like the convolutional layer, pooling involves moving a kernel across the input however unlike the convolutional layer this kernel lacks weights. The kernel applies an aggregation function to the values within its receptive field, filling up the output array. There are two main types of pooling: Max Pooling and Average Pooling. Max Pooling selects the highest value within the portion of the input covered by the kernel, acting as a noise suppressant and contributing to denoising alongside dimensionality reduction. Average Pooling returns the average of the values within its covered area also serving as a mechanism for noise suppression and dimensionality reduction. While both pooling methods serve to streamline the information Max Pooling tends to outperform Average Pooling, particularly in managing noise and maintaining efficiency in the neural network.

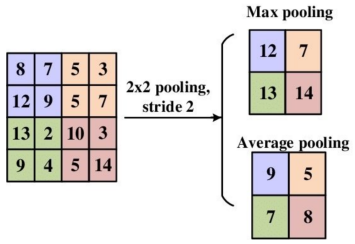


FIGURE 2.5: Application of max and average pooling [28]

2.4.3 Fully Connected Layer

A full-connected layer unlike partially connected layers connections between input and output nodes are selective in the fully-connected layer there is a direct link between every node in the output layer and each node in the preceding layer. This layer performs the task of classification based on the features extracted by the previous layers and their different kernels.

In traditional architecture after the image has been passed through all the convolutional and pooling layers it is flattened into a column vector. The flattened output is then linked to one or more fully connected layers, forming the neural network's concluding stage. Over a series of epochs of training the model learns to discern dominant and specific low-level features within images. The final layer is a softmax classification layer

2.4.4 Global Average Pooling

In more modern architectures instead of fully connected layers Global Average Pooling [29] is used. Global Average Pooling is an operation that calculates the average output of each feature map in the previous layer. Instead of adding fully connected layers on top of the feature maps the average of each feature map is calculated, and the resulting vector is fed directly into the Softmax layer. Global Average Pooling significantly reduces the number of parameters in the network, making it computationally more efficient.It also introduces a form of spatial hierarchies – by taking the average, the network focuses on the overall importance of features rather than specific details, which increases robustness and reduces overfitting.

2.5 Transformers

A transformer model[30][31] is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in a sentence. These models have been applied to a wide range of tasks in machine learning and artificial intelligence, significantly advancing the capabilities of automated systems. The model was first described in the 2017 paper "Attention is All You Need" [32]. The architecture of a transformer model is composed of an encoder-decoder structure, where both the encoder and decoder consist of a stack of identical layers. Each layer contains two main components: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network.

- **Multi-Head Self-Attention Mechanism:** This allows the model to attend to different positions of the input sequence simultaneously, capturing various aspects of contextual relationships.
- **Feed-Forward Neural Network:** A fully connected network that processes the output of the self-attention layer, applying non-linear transformations to capture complex patterns.

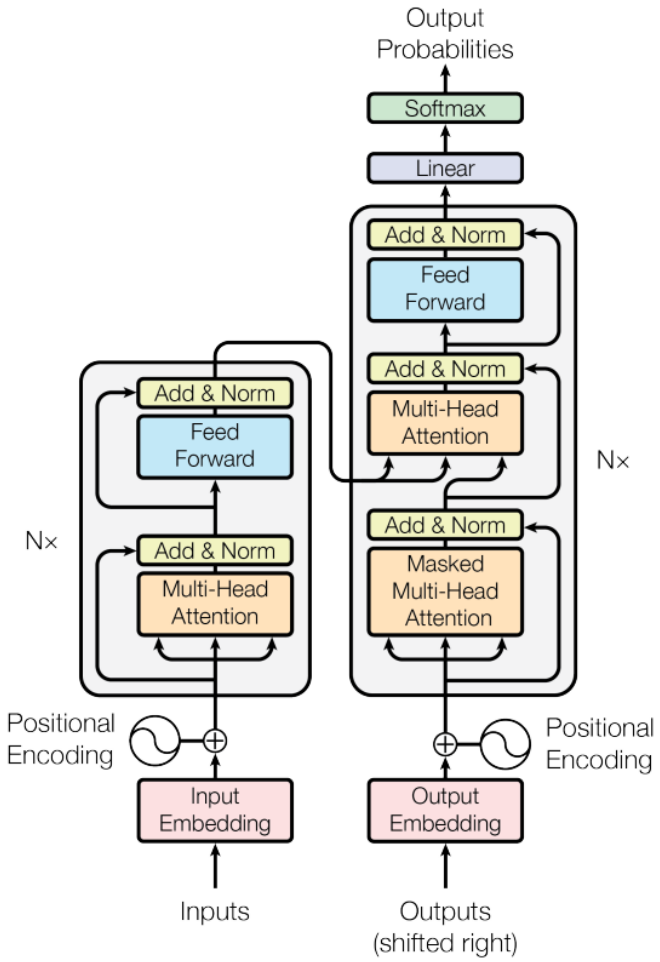


FIGURE 2.6: The architecture of the transformer model as designed in the original "Attention is All You Need" paper[32].

One of the primary innovations in transformer models is positional encoding. Instead of processing each word in the order that it appears, a unique number is assigned to each word, providing information about the position of each token (parts of the input such as words or subword pieces in NLP) in the sequence. This allows the model to incorporate the sequential information of the input data, enhancing its ability to understand context and relationships within the sequence. Another innovation of the transformer models is the self-attention mechanism. It computes a set of attention weights for each word in the input sequence, allowing the model to focus on relevant parts of the sequence when making predictions.

Transformer models offer several advantages over previous architectures such as parallelization: Unlike RNNs, which process sequences sequentially, transformers allow for parallel processing of sequence elements, significantly speeding up training and inference. Scalability: The self-attention mechanism enables transformers to handle very long sequences, making them suitable for tasks like document summarization and machine translation. Efficiency: Transformer models require less

training time than previous recurrent neural network architectures such as long short-term memory (LSTM) networks, which process sequences in a step-by-step manner.

More recently in addition to natural language processing tasks, transformers have also been successfully applied to computer vision tasks. Vision transformers (ViTs) adapt the transformer architecture for image processing, enabling models to handle image classification, object detection, and other vision-related tasks with remarkable efficiency and accuracy[33].

Chapter 3

Related Work

To the best of our knowledge there hasn't been any work done on Hallucination detection in image inpainting. We present below however work done in image inpainting, Remote Sensing Image Classification, Out of distribution detection as well as some general hallucinations research.

3.1 Image Inpainting

Image inpainting methods consist of the traditional approach while recently with the advent of Generative Adversarial Networks (GANs) and Convolutional Neural Networks (CNNs) new methods Deep Learning Methods have been developed. Traditional methods include the following Exemplar-based texture synthesis methods [34][35]which uses patches from the undamaged portions of the image to fill in the missing areas. Exemplar-based Structure Synthesis methods [36] [37] which while similar to texture synthesis uses statistical constraints to sample patches. Diffusion-based Methods [38][39]which propagate pixel information from the boundary into the missing regions. Sparse Representation Methods[40][41] which represents the image as a sparse linear combination of elements from a dictionary and fills in the missing regions with patches that have a similar sparse representation to the known areas. Hybrid Methods [42][43]which are a combination of texture and structure synthesis.

The deep learning methods make use of CNNs and GANs. Convolutional Neural Networks (CNNs)[44][45] CNNs learn features from images making them effective for image inpainting. Generative Adversarial Networks (GANs)[46][47][48] GANs consist of a generator and a discriminator that compete against each other to try and make more realistic looking images leading to more realistic inpainting results.

3.2 Remote Sensing Image Classification

Automatic satellite image classification falls into two categories Supervised and Unsupervised learning[49]. Supervised learning methods require human input for training, among others Supervised learning methods include Artificial Neural Networks which associate labels to data, Binary Decision Trees which contain binary rules that determine the classification of individual pixels and Image Segmentation entail grouping pixels that exhibit relative homogeneity into segments so that classification of segmented images can occur at the segmentation level.

Unsupervised classification methods make use of clustering algorithms to group image pixels into unlabeled classes which are subsequently manually assigned labels. The main unsupervised classification methods are Self-Organizing Data Analysis Technique (ISODATA), Support Vector Machines and K-Means. The ISODATA algorithm [50] iteratively partitions a dataset into a specified number of clusters based on the statistical properties of the data while dynamically adjusting the number of clusters during the process. K-means is a clustering algorithm that groups data points into K clusters by minimizing the distance between each point and the centroid of its assigned cluster. The SVM algorithm [51] works by finding the best hyperplane that separates different classes in the feature space while maximizing the distance between the hyperplane and the nearest data points of each class.

Recent advancements explore applying reinforcement learning (RL) to remote sensing image classification, marking a shift towards more adaptive and autonomous systems. RL methods, traditionally used in dynamic decision-making tasks, are being adapted to optimize classification processes by dynamically learning from interactions with image data. RL frameworks can enhance classification accuracy by iteratively refining classification decisions based on feedback received during training, potentially offering robust performance in complex and evolving remote sensing environments [52].

For instance, the Scale-Aware Classification Network (SAN) introduces a deep reinforcement learning-based approach to automatically select the optimal scale of image patches for land-use/land-cover classification from high spatial resolution remote sensing imagery. Using a scale control agent and feature indexing module SAN adapts network parameters to effectively handle different-scale objects within images, demonstrating superior performance compared to traditional fixed-patch methods [53].

Similarly, the Deep Q-Fully CNN (DQFCN) framework addresses challenges in PolSAR image classification by integrating RL to optimize feature extraction from spatial-polarimetric data and improve classification accuracy with limited labeled samples. This approach represents a significant advancement in utilizing RL for remote sensing tasks, showcasing enhanced performance across multiple benchmark datasets [54].

3.3 Hallucinations

There hasn't been any work on hallucination detection in image inpainting to the best of our knowledge but there are some works that are closely related like Spotting Hallucinations in Inverse Problems with Data-Driven Priors[55]. In that work the authors reconstruct galaxy images using generative networks and by computing the diagonal elements of the Fisher information matrix for the likelihood and the data-driven prior separately they identify regions that are prone to hallucinations or incorrect realistic looking features. In [56] tomographic images are reconstructed via data driven (U-NET) and non data driven (PLS-TV, DIP) methods from undersampled data. They measure hallucinations through hallucination maps which compare the reconstructed images with the ground truth and with metrics such as Mean Square Error (MSE) and Structural Similarity Index (SSIM) to assess the fidelity of the reconstructed images. In another computer tomography hallucination study [57] the authors simulate structure embedded noise and train a model to recognize and quantify spurious structural components or hallucinations in the noise. To measure hallucinations in residual noise they use metrics such as Residual Structure Index (RSI) which is calculated by processing the residual noise through the model to predict structural components within it. As well as Heatmap Visualization, the heatmaps are created by superimposing the absolute value of the predicted output onto the input residual image localizing therefore spurious structures or hallucinations.

3.4 Out of Distribution Detection

There have been two notable works on Out of Distribution Detection (OOD) that make use of a generator in order to reconstruct a new image from a damaged ground Truth image.

Liu et al(2023)[58] proposed Lift, Map, Detect (LMD), an unsupervised out-of-distribution (OOD) detection method that uses diffusion models. LMD operates under the assumption that a diffusion model trained on in-domain data can effectively map a

corrupted in-domain image back to its original state, while struggling to do the same for out-of-domain images. The method first lifts an image off its original manifold by masking parts of it. Then, it maps the lifted image towards the in-domain manifold using a diffusion generator model trained on in-domain data. The distance between the original and mapped images is calculated using the LPIPS metric and is then used as an OOD score with larger distances meaning a higher likelihood of the image being out-of-domain.

Graham et al. (2023)[59] introduce a novel approach to unsupervised out-of-distribution (OOD) detection that leverages the unique properties of denoising diffusion probabilistic models (DDPMs). DDPMs, typically used for image generation, are reinterpreted as denoising autoencoders with an externally controllable bottleneck. This bottleneck is manipulated by varying the amount of noise added to the input image. The method, referred to as DDPM-based OOD detection it involves reconstructing the input image from multiple noise levels and assessing the reconstruction error at each level. The result is a multi-dimensional reconstruction error that is used to determine if the input image is in-distribution or out-of-distribution. The authors demonstrate the effectiveness of their approach on both standard computer vision datasets and higher-dimensional medical imaging datasets, showcasing superior performance compared to existing generative and reconstruction-based OOD detection methods. Notably, the DDPM-based method eliminates the need for dataset-specific tuning, a common issue in reconstruction-based approaches, by automatically incorporating results from a range of noise bottlenecks.

Chapter 4

Method

4.1 Overall architecture

As can be seen in Fig.4.1 The proposed hallucination detection architecture consists of an image generator that fills the missing parts of an image and a classifier responsible for discriminating between a realistic and a non-realstic image. The generator is trained on masked examples of images from a specific dataset while the input image is selected from another dataset. The classifier is trained on the 2 aforementioned datasets and is responsible for assigning a probability that the output image from the same class is used to train the generator or not. Furthermore, a class activation mapping (CAMs) process using the Grad-Cam model, is also introduced in order to shed light on the decision-making process of the classifier. Finally, the CAMs are thresholded and the dice metric is calculated between the thresholded CAMs and the mask of the image in addition to the probability distribution calculated by the classifier.

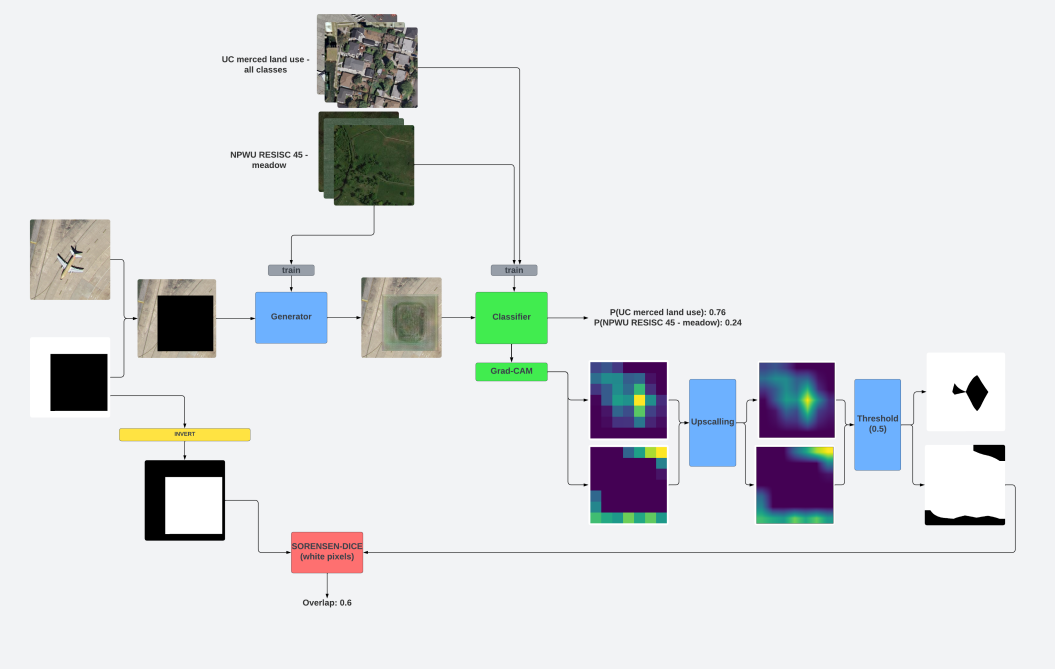


FIGURE 4.1: Combined architecture

4.2 Generator

We employed the Mask-Aware Transformer for Large Hole Image Inpainting (MAT) as described in the work [60] with MAT being a transformer-based model for large hole inpainting. The role of our Generator is to inpaint the missing regions of the masked image I_M which can be described as $I_M = I \odot M$ with I being the ground truth image and M being the mask with $M = 0$ for the missing pixels and $M = 1$ for other pixels and $\odot$ being the Hadamard product. The generator therefore aims to create a reconstructed realistic image I_G. The generator G consists of a convolutional head, a transformer body, a convolutional tail, a style manipulation module and a Conv-U-Net, as shown in Figure 4.2.

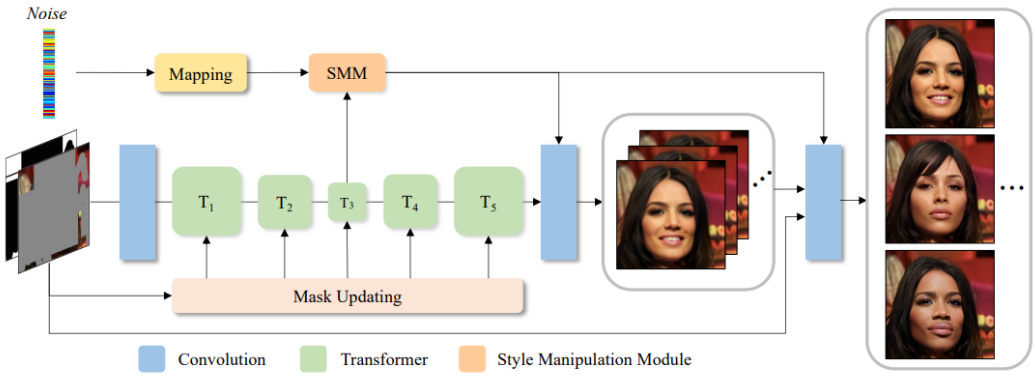


FIGURE 4.2: MAT architecture [60]

4.2.1 Convolutional Head

The convolutional head consists of four convolutional layers and processes the incomplete input image I_M and mask M to generate lower-resolution feature maps ($\frac{1}{8}$ the size of the original image). The feature maps are used to create tokens that represent the image's content.

4.2.2 Transformer Body

The transformer body consists of 5 transformer blocks. Each transformer block concatenates the input and output of attention while using a fully connected layer as can be seen below:

$$\begin{aligned} X'_{k,l} &= FC\left(\left[MSA\left(X_{k,l-1}\right), X_{k,l-1}\right]\right) \\ X_{k,l} &= MLP\left(X'_{k,l}\right) \end{aligned} \tag{4.1}$$

The output of the MLP module of the l-th block in the k-th stage is denoted as $X_{k,l}$. After several Transformer blocks a convolutional layer with global residual connection is added.

4.2.3 Convolutional Tail

The convolutional tail is responsible for upsampling the feature size from $\frac{H}{8} \times \frac{W}{8}$ back to $H \times W$, thereby generating a completed image. During this process, style modulation is applied to all layers, facilitating pluralistic generation.

4.2.4 Style Manipulation Module

The style manipulation module applies style representations conditioned on both the input image and noise, modulating the features throughout the network. This helps in achieving diverse and realistic inpainting results.

4.2.5 Conv-U-Net

The Conv-U-Net serves as the second stage in the MAT framework, refining the coarse predictions made by the first stage. It takes the output image and mask as inputs, first downsampling the feature size to $\frac{H}{32} \times \frac{W}{32}$, and then upsampling it back to the original size $H \times W$. Shortcut connections at each resolution level facilitate the retention of high-level features. The number of convolution channels in the encoder starts at 64 and doubles with each downsampling step, peaking at 512, with

a symmetrical configuration in the decoder. This structure enhances the network's capacity to generate high-fidelity details.

4.2.6 Generator Loss functions

The loss function relies on non-saturating adversarial loss, R1 regularization or $R1 = E_x ||\nabla D(x)||$ as well as perceptual loss with a low coefficient for easier optimization. The adversarial Loss is calculated as follows

$$\begin{aligned} L_G &= -\mathbb{E}_{\hat{x}}[log(D(\hat{x}))], \\ L_G &= -\mathbb{E}_{\hat{x}}[log(D(\hat{x}))] - \mathbb{E}_{\hat{x}}[1 - log(D(\hat{x}))], \end{aligned} \tag{4.2}$$

With x and $\hat{x}$ being the real and generated image respectively.

The perceptual Loss can be articulated as

$$L_P = \sum_i \eta_i \left|\phi_i(x) - \phi_i(x)\right|_1 \tag{4.3}$$

With s the layer activation of a pre-trained VGG-19 network being denoted as $\phi_i(\cdot)$

The overall loss is formulated as

$$L = L_G + \gamma R_1 + \lambda L_P. \tag{4.4}$$

With γ and λ equal 10 and 0.1 respectively.

4.3 Classifier

The Classifier C receives the inpainted image I_G and being trained on 2 datasets creates 2 probabilities p_1 and p_2 for the image I_G belonging to each of the 2 datasets.

The Classifier was constructed through Transfer Learning, utilizing the MobileNetV2 model [61] so on top of the base model we introduced a GlobalAveragePooling2D layer. The model transforms the images from a 224x224x3 array into a compact representation consisting of a 7x7x1280 block of features. Subsequently, the GlobalAveragePooling2D layer converts each block of features into a 1280-dimensional

vector.

$$GAP(X)_c = \frac{1}{\mathrm{H} \times \mathrm{W}} \sum_{i=1}^{\mathrm{H}} \sum_{j=1}^{\mathrm{W}} X_{ijc} \tag{4.5}$$

$GAP(X)_c$ is the output of the GlobalAveragePooling2D operation for channel c while X_{ijc} presents the value of the input feature map X at spatial location (i, j) and channel c. In our case, the height H and width W are both equal to 7.

We then add a Dropout layer with dropout rate $p = 0.5$ on top of the GlobalAveragePooling2D layer to prevent overfitting which can be modeled as

$$Dropout(x) = \begin{cases} 0, & \text{with probability } p = 0.5 \\ \frac{x}{1-p}, & \text{otherwise} \end{cases}$$

where x is the input to the dropout layer.

Finally adding a Dense layer on top of that turns the 1280 vector into a 2-dimensional output, effectively performing classification between 2 classes.

4.4 Grad-CAM

Grad-CAM (Gradient-weighted Class Activation Mapping)[62] works based on the fact that convolutional layers retain spatial information that is lost in the fully connected layers, it can therefore be expected that the last convolutional layers of a network have the best compromise between high-level semantics and detail spatial information. The neurons in the last layers look for semantic class-specific information in the image.

Grad-CAM works utilizes the gradients of any target concept, flowing into the final convolutional layer to produce a localization map highlighting the regions that the model considers important for predicting the concept. To obtain the class-discriminative localization map *Grad-CAM* $L_c^{\text{Grad-CAM}} \in \mathbb{R}^{u \times v}$ with width u and height v for any class c gradient of the score for class c, y^c (before applying softmax), is computed with respect to the feature map activations A^k of a convolutional layer, denoted as $\frac{\partial y^c}{\partial A^k}$. These gradients are then global-average-pooled over the width and height dimensions (indexed by i and j respectively) to obtain the neuron importance weights α_k^c:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{4.6}$$

While computing the importance weights α_k^c during the backpropagation of gradients with respect to activations, the exact process involves successive matrix multiplications of the weight matrices and the gradient with respect to the activation functions until reaching the final convolution layer being backpropagated to. Therefore, α_k^c represents a partial linearization of the network from A onwards, capturing the 'importance' of feature map k for the target class c. A weighted combination of the forward activation maps is then applied along with a ReLU to obtain the final map.

$$L_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right) \tag{4.7}$$

The ReLU is applied to the linear combination of maps because only the features that have a positive influence on the class is of interest.

In each experiment, We apply Grad-CAM to the classifier for added model explainability. Grad-CAM is applied by creating a new model with the same inputs as the classifier but with its output set as the classifier's last convolutional layer. The Grad-CAM model can produce two 7x7 pixels feature maps one for each class which upon up-sampling to the input image size generates two heatmaps.

4.5 Masks

The generator is fed images and masks, both sized at 512x512 pixels. The masks representing ground truth are binary with white denoting areas to be preserved and black indicating regions to be masked out. Various mask shapes and sizes are used in the inpainting process covering a size range from occupying 0% of the image to covering 90% of the image or 235.929 pixels and everything in between with a step of 10%

Three distinct mask shapes are used: rectangular, random, and irregular one instance of those can be seen in Fig. B.6 denoted as Rect, Ran, and Ir accordingly. Rectangular masks are introduced as randomly positioned rectangles within the canvas borders, with subsequent mask sizes also placed randomly. Random masks select pixels across the canvas randomly for masking, with each subsequent mask following suit. Irregular masks use the flood-fill algorithm starting from the center of the image and expanding by adding adjacent pixels randomly to the last added pixel.

The irregular masks aim to be somewhat of a middle ground between the rectangular and the random pixel masks having sometimes solid masked areas like the rectangular masks and other times more fluid random masked areas.

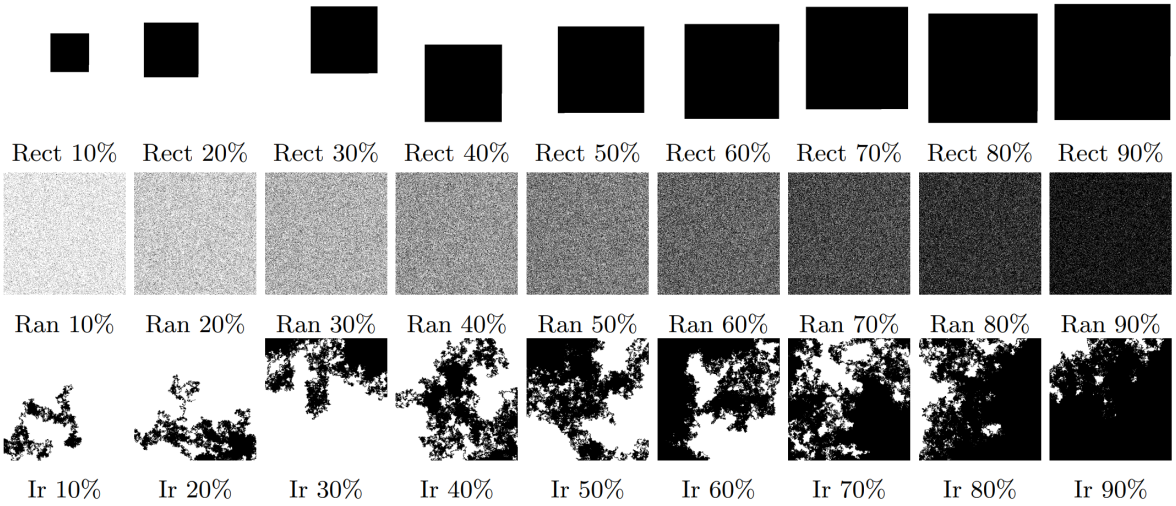


FIGURE 4.3: Instance of Masks

4.6 Data and training

4.6.1 Generator

We employ a generator that has been pre-trained on the CelebA-HQ_256 dataset which consists of 30.000 high-quality images of celebrity faces. For remote sensing experiments, we trained the generator on parts of the NWPU-RESISC45 dataset. The NWPU-RESISC45 dataset is a publicly available benchmark for Remote Sensing Image Scene Classification (RESISC), created by Northwestern Polytechnical University (NWPU) and it contains 31,500 images among 45 classes with 700 Images in each class. The generator was trained in 300 images sized 512x512 in .png format at a time from each class as the rest were used for testing and training the classifier.

To evaluate each model's performance, we used 10 images with the same rectangular mask and through inference calculated the Fréchet inception distance (FID) with pytorch [63]. When the FID metric converged to a set value we stopped the training. Below can be seen the FID values for the training process of a NPWU-RESISC45 meadow and a NPWU-RESISC45 dense residential generator respectively

Indicatively some instances of the generator's training process can be seen in Fig.4.5 for the meadow trained generator and in Fig.4.6 for the dense residential trained generator. Additional results can be found in Appendix B.

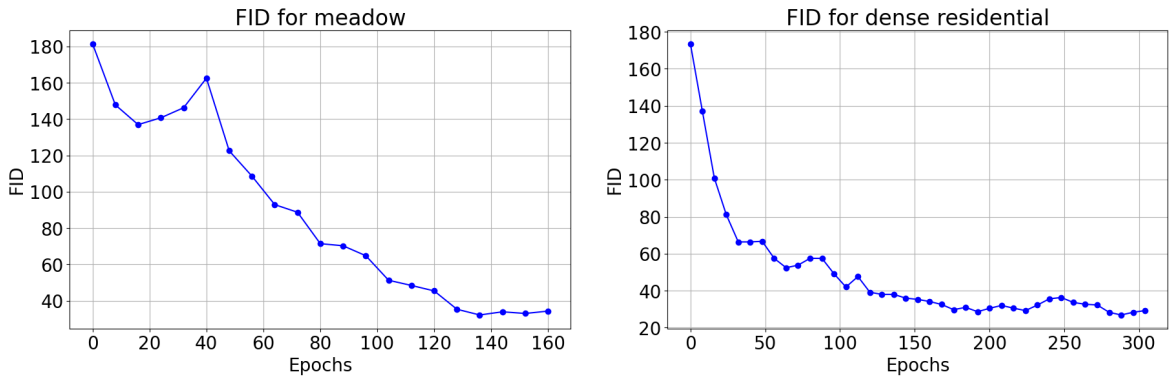


FIGURE 4.4: RESISC45 meadow and dense residential class FID

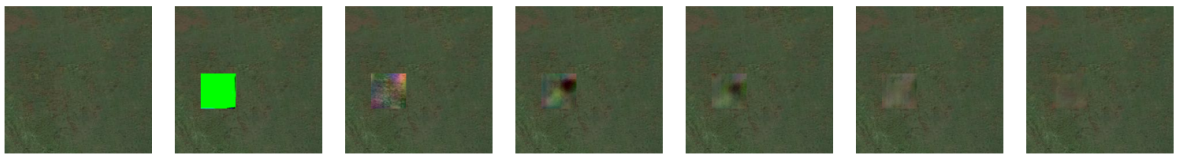


FIGURE 4.5: Generator Training instances for RESISC45 meadow



FIGURE 4.6: Generator Training instances for RESISC45 dense residential

4.6.2 Discriminator

The Discriminator was constructed through Transfer Learning, utilizing the MobileNetV2 model [61], pre-loaded with weights trained on the ImageNet dataset [64], a widely-used research training dataset encompassing a broad array of visual categories. The reason behind using transfer learning was to save computing resources, as a model pretrained on such an extensive and diverse dataset inherently possesses a foundational understanding of the general visual world.

The classifier training datasets differ depending on the specific experiment. The NWPU-RESISC45 dataset was used which consists of remote sensing satellite photographs while for single-class training, one class of the NPWU-RESISC45 dataset was used to represent the in-distribution class, while the rest of the classes were used to represent the out-of-distribution examples.

As a pre-processing step, the images were resized to dimensions of 224x224 since that is the highest dimension MobileNetV2 allows. The images also underwent

the necessary preprocessing for the MobileNetV2 which includes turning their data type to float32 array and scaling pixel value to [-1,1]. Subsequently, the preprocessed images were batched in groups of 32 and provided as input to the model.

The model was initialized without the classification layers which are specific to the original class labels it was initially trained on. For the purpose of feature extraction, the convolutional base was frozen preventing any modifications to the weights within individual layers during the course of training. On top of this frozen convolutional base the aforementioned layers were added and consequently, only those underwent training throughout the training process.

By introducing these additional layers, our model's architecture resulted in a total of 2.562 trainable parameters. This contrasts significantly with the original model's architecture which would have entailed the training of 2.260.546 total parameters had we not leveraged transfer learning.

4.7 Experimental Procedure

4.7.1 Generator Testing

For each experiment 10 different images of a specific class or set of classes are selected the images belong to either the NPWU-RESISC45 dataset or the UC Merced Land Use Dataset. Ten iterations of each mask type are applied to each image for a total of 30 masks. Once the inpainting procedure is completed for every image the metrics MSE, FID, LPIPS are used to evaluate the inpainting accuracy of the generator.

Mean Square Error - MSE

The Mean Squared Error (MSE) calculates the squared difference between the predicted values in our case pixels $\hat{y}$ and the actual pixel values y across all data points divided by the number of samples $n_{samples}$ so as to get the mean value. The formula for MSE can be seen below:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2.$$

The MSE was calculated between grayscale images since what is of interest to us is the semantic content and not the color difference which can skew the MSE result by a huge margin.

Fréchet inception distance - FID

We use FID to quantitatively assess the fidelity of generated images. The FID [65] compares the feature-wise distribution of generated images to that of the ground truth images measuring in essence the Fréchet distance between two multivariate Gaussian distributions.

For the calculation of FID a separate pre-trained neural network, the Inception-V3 is used to extract features from the generated and real images. These features are assumed to follow a multivariate Gaussian distribution. For the real images, the mean m_w and the covariance C_w of the extracted features are computed, and similarly, the mean m and covariance C of the extracted features are calculated for the generated images. The FID is computed as

$$d^2((m, C), (m_w, C_w)) = \|m - m_w\|_2^2 + \mathrm{Tr}(C + C_w - 2(CC_w)^{1/2}) \tag{4.8}$$

In the above equation, $||m - m_w||_2^2$ represents the squared Euclidean distance between the means of the two distributions, capturing the difference in the central tendencies of the feature distributions. The term $\mathrm{Tr}(C + C_w - 2(CC_w)^{1/2})$ is the trace of the matrix, which measures the sum of the diagonal elements. This term captures the differences in the covariances of the two distributions, reflecting the spread and shape of the feature distributions. The matrix square root $(CC_w)^{1/2}$ ensures that the covariance differences are measured appropriately.

Learned Perceptual Image Patch Similarity - LPIPS

The Learned Perceptual Image Patch Similarity (LPIPS) metric[66] is used to quantify the perceptual similarity between two image patches. A pre-trained neural network is used in our case AlexNet was used though LPIPS can also employ SqueezeNet and VGG to capture complex image structures that traditional metrics like PSNR and SSIM often miss. LPIPS is particularly effective in scenarios involving spatial ambiguities, where classic metrics fall short.

LPIPS operates by computing distances in the feature space of a deep network. Given two image patches, x and x_0, and a network F, LPIPS extracts feature stacks from L layers of the network. The activations are then normalized across the channel dimension, resulting in $\hat{y}_l$ and $\hat{y}_{l_0}$ for each layer l. Each channel is then scaled by a learned weight vector w_l, and the ℓ_2 distance is computed. This process is averaged

spatially and summed across channels. The LPIPS equation can be seen below:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}_{lhw} - \hat{y}_{l_0hw})||_2^2 \tag{4.9}$$

Where $\odot$ is the Hadamard product or element-wise multiplication.

The weights w_l can be learned through various configurations. In the "linear" configuration (LPIPS-lin) the pre-trained network weights are frozen, and only the linear weights are learned effectively calibrating the existing feature space. In the "tune" configuration (LPIPS-tune) the entire network is fine-tuned on perceptual similarity judgments. While in the "scratch" configuration (LPIPS-scratch) the network is trained from scratch on the perceptual similarity task. For our experiments, the weights were set to "linear" configuration

4.7.2 Discriminator Testing

Similar to the generator testing procedure, the classifier testing also utilized an equivalent number of masks and images. In the experiments detailed in Section 5.1 and Section 5.2, 10 different images were used, each paired with 10 iterations of all three mask types. Meanwhile, for the experiments in Section 5.3, 5 different images were used, each with 5 iterations of all three mask types.

Once the inpainting procedure is completed for all images, they undergo evaluation using both the classifier and the Grad-CAM model. The classifier outputs probabilities indicating the likelihood of the image belonging to each class it has been trained to recognize p_1 and p_2, while the Grad-CAM model generates two class activation maps which upon upscaling transform into heatmaps. They are then thresholded with a threshold value of 0.5 to produce the thresholded image.

$$THmap(i,j) = \begin{cases} 1, & Hmap(i,j) < 0.5 \\ 0, & Hmap(i,j) \geq 0.5 \end{cases}$$

The evaluation metrics are the classifier's probabilities alongside the Sørensen-Dice coefficient computed between the mask and the thresholded heatmaps which can be formulated as:

$$DSC = \frac{2 \cdot |M \cap THmap|}{|M| + |THmap|} \tag{4.10}$$

where M is the mask and $THmap$ is the thresholded heatmap. It is noteworthy that when calculating the dice coefficient for out-of-distribution (OOD) heatmaps the masked image is color-reversed so as to be color-matched with the $THmap$. There are numerous combinations where the Dice coefficient can be applied, including comparisons between masks and thresholded heatmaps, as well as considering intersections of white, black, or total pixels. All these combinations were tested. The most effective combination, which closely matched the y=x line representing the increasing rate of hallucinations with increasing mask coverage, was the Dice metric between the thresholded in-distribution (ID) heatmap and the inverted mask, accounting for white pixels and the OOD prediction value of the other metrics of some of the experiments can be seen in the Appendix A. For the dice metric combination that was selected, the Sørensen-Dice coefficient measures the spatial absence of the in-distribution (ID) ground truth image from the inpainted image as can be detected by the discriminator. Refer to Figure 4.7 for an illustration of the heatmaps and corresponding thresholded heatmaps generated by the discriminator, showcasing variations across different masked areas.

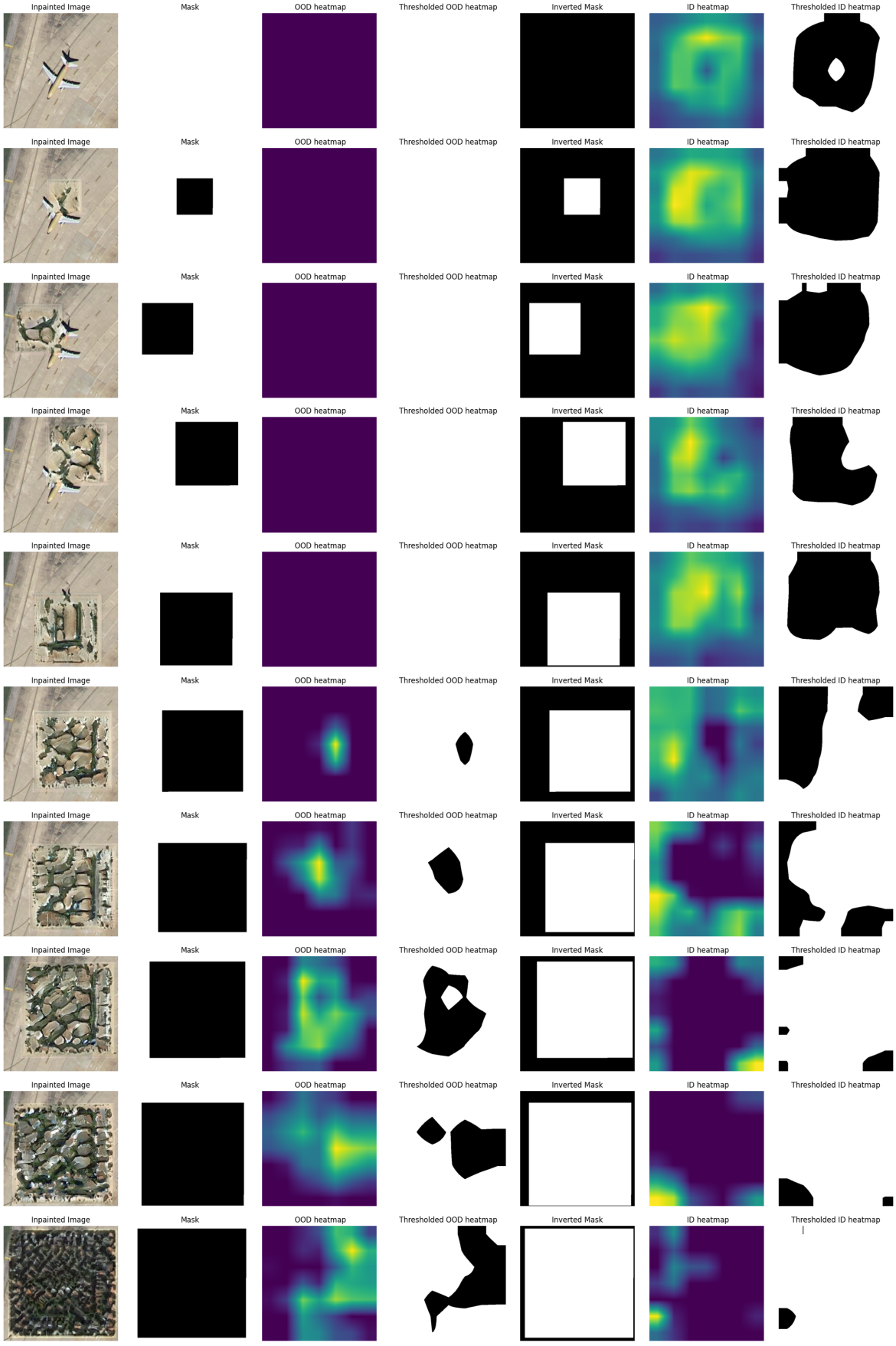


FIGURE 4.7: Inpainted results, heatmaps, and thresholded heatmaps for a rectangular mask with a generator trained on dense residential and ground truth belonging to an airplane. Each mask is 10% image-wise bigger than the previous

Chapter 5

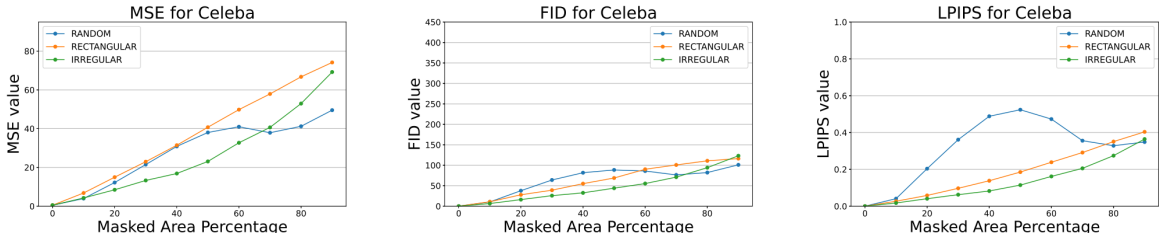
Experimental Results and Analysis

This section's experiment aim to test and compare the generators reconstruction ability using three metrics MSE, FID and LPIPS. The experiments include a pre trained generator trained by the original paper's authors and two generators trained by us from scrath based on the architecture of the pretrained generator.

5.1 Generator Testing

5.1.1 CELEBA trained Generator Test on CELEBA input

In this experiment we analyse the MAT generator model, pre-trained on CELEBA data and applying the aforementioned metrics between the inpainted image or output of the generator and the ground truth image. The images to be inpainted by the generator also belong in the CELEBA dataset this can therefore be considered the optimal scenario for the generator.



(a) MSE metrics (b) FID metrics (c) LPIPS metrics

FIGURE 5.1: MSE,FID, LPIPS measurements for generator trained on Celeba

As for the MSE metrics it can be seen that for every single percentage of masked area tested the rectangular mask produces the greatest MSE compared to the other two masks random and irregular. While for the rectangular mask the MSE value is increasing steadily which doesn't seems to be the case for the random and irregular masks, with the irregular mask starting off with a steady increase lower than that of rectangular but as the masked area increases the MSE value is increasing

exponentially. The random mask seems to be increasing with the same rate as the rectangular while it slows off and even drops after 60% mask coverage thus creating a bulge before increasing again in the last masked areas.

The FID plot has a similar dynamic between the rectangular and irregular masks with the rectangular mask's FID always above the irregular mask's FID and the rate of increase being very similar to the one in the MSE plot with the slight difference that the exponential increase of the irregular mask not being as pronounced as in the MSE plot. The Random mask's plot same as in the MSE forms a bulge this time surpassing the rectangular mask's amplitude and then falling behind it after the peak of the bulge.

Finally the LPIPS plot once again has a similar dynamic for the values of the rectangular and irregular masks while the random mask still maintaining a bulge around 50% coverage has a significantly larger LPIPS amplitude compared the other two masks and the previous plots.

Overall the behaviour of the rectangular and irregular mask's reconstruction is playing out as expected with the metrics portraying a worsening reconstruction as the masked area increases. The random mask's reconstruction however doesn't follow the same logic suggesting a significantly worsening reconstruction starting at 30% or 40% and ending in around 70% mask coverage and then worsening linearly after that threshold.

If we are to examine a random instance of inpainting in Fig.5.2 we can see that for the random mask starting from as soon as 30% though to a limited degree and up to 70% again in a limited degree the inpainted image starts to jitter. This phenomenon is the most acute at around 50% and can be easily spotted especially when observing the background of the face where the uni color is broken by the random pixel mask inpainting which introduces pixels that are not coherent with their surroundings with colors that seems to be either very close to white or very close to black. At very big masking areas with random masks such as 90% the reconstructed pixels do not have this variance and color discrepancy however since very little original information is present the inpainted image is similar to the original though blurry since the remaining ground truth information of the image is more or less evenly distributed across the whole image therefore the generator only has to fill in between. However small features such as eyes , eyelids and teeth have some mismatched inpainted pixels since these features are have a high degree of variance (not as uniform as skin) in a small space.

It can also be inferred from the graphs that the rectangular masks offer an overall worse reconstruction in comparison to the irregular masks especially for medium masked areas while they are more comparable at the bigger masked areas. This is the case since the rectangular masks damage a solid part of the image and the generator is left to reconstruct it with limited context in contrast to irregular shaped masks which offer a middle ground between rectangular and random masks masking some solid parts of the image and in others leaving more context due to their chaotic nature. It can be seen in Fig.5.2 as well that the reconstructed results of the irregular shaped image are more faithfull to the ground truth image compared to those of the rectangular mask and only in very big masked areas (90%) do they start to deviate, in the case of Fig.5.2 inpainting a different haircut, while in rectangular masks significant deviations can be seen earlier such deviations include gender change or other different facial features.

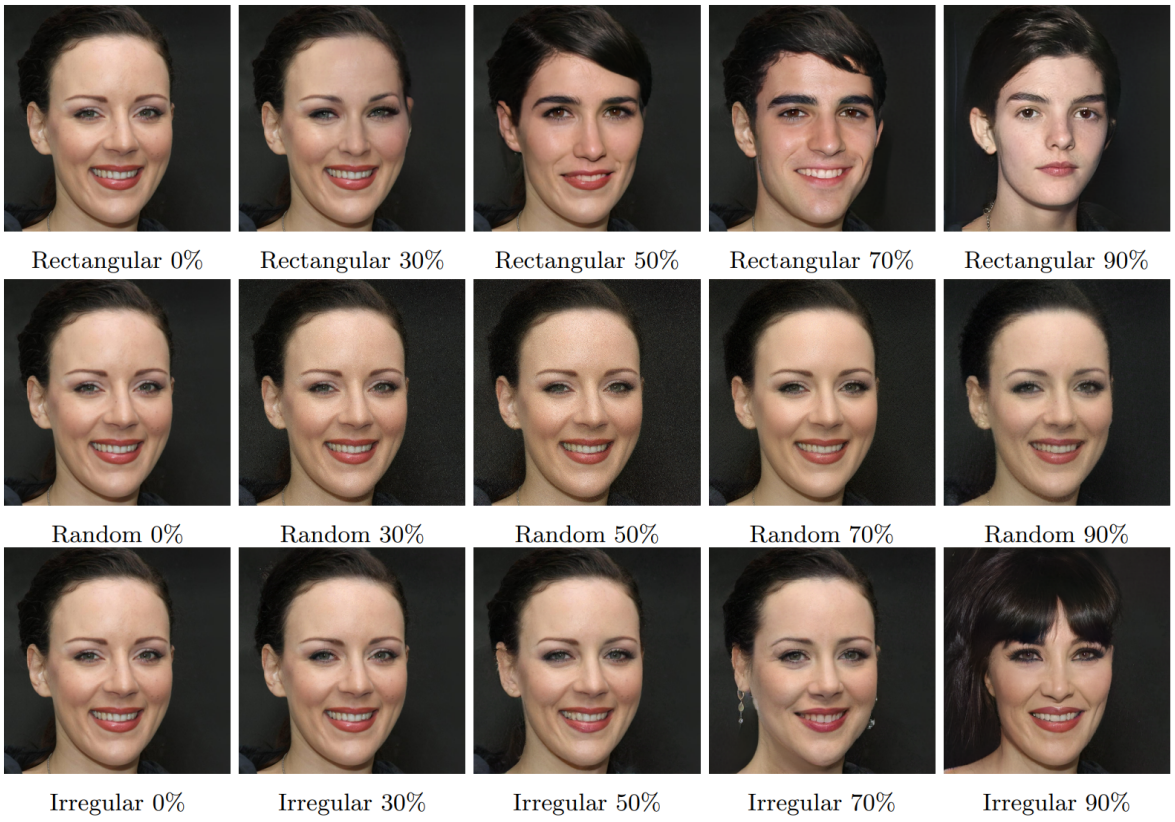


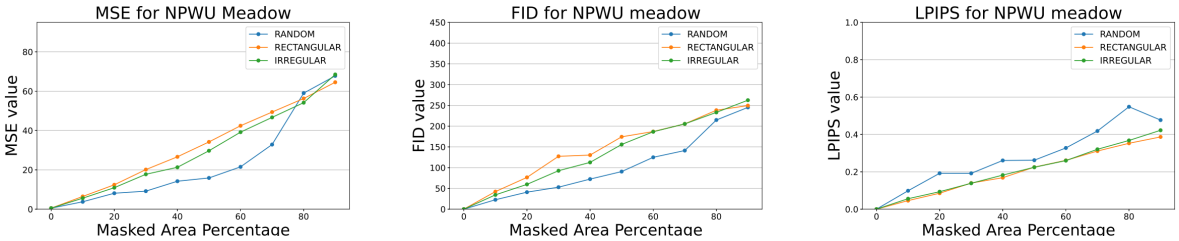
FIGURE 5.2: Celeba inpainted results

5.1.2 RESISC45 Meadow trained Generator Test on RESISC45 meadow input

In this experiment we a custom MAT generator trained from scratch on the NPWU-RESISC45 dataset's Meadow class and applied the aforementioned metrics between the inpainted and truth images. The images to be inpainted by the generator also belong in the NPWU-RESISC45 dataset's Meadow class. Some examples are shown in Figure XX



FIGURE 5.3: Examples from the meadow class.



(a) MSE metrics (b) FID metrics (c) LPIPS metrics

FIGURE 5.4: MSE,FID, LPIPS measurements for generator trained on RESISC45 Meadow

As can be seen from the MSE metrics plot the rectangular masks offer the worst reconstruction MSE wise in all instances except for very large masked patches. Irregular masked images are a close second which along with rectangular masks are increasing their MSE values linearly with the mask size. The lowest MSE values are offered by the random masks which especially for mask sizes between 40% and 70% have a noticable difference in MSE value between them and the other two masks before converging to the same MSE value as the others in 80% and 90% maksed patches. With the exception of random masks which saw a curbe in the Celeba case the MSE values of the Celeba and Meadow experiments are very similar.

In the FID plot the rectangular masks seems to be slightly worse that irregular masks

and the random masks are the best with a significant difference, all FID measurements are linearly increasing together with the masked area percentage. It is also apparent that both the amplitude and the rate of increase of the FID is higher in the meadow case compared to the Celeba case.

The LPIPS plot is very similar with the Celeba LPIPS plot with the only difference being that rectangular and irregular masks practically overlap and that the random mask does not have a bulge shape but is linear and has a higher value than the rest of the masks.

As can be seen in Fig.5.5 the inpainting results are as expected the random mask inpainting produces a blurry reconstruction of the ground truth image while the rectangular masks do the worst reconstruction missing details of the original image due to the nature of the rectangular mask which may mask some details in their entirety leaving no chance for reconstruction of them. Irregular again acts as somewhat of a middle ground between the random and rectangular masks it may or may not cover details in their entirety.

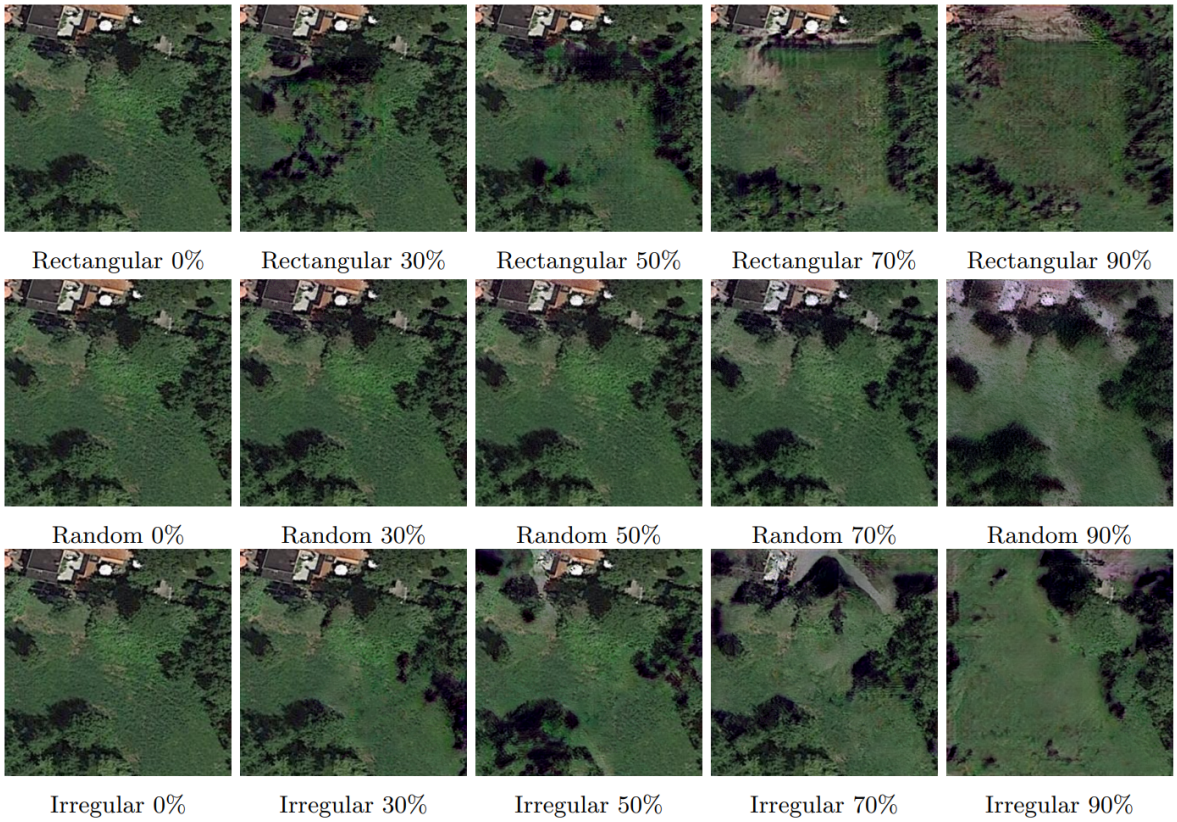


FIGURE 5.5: RESISC45 meadow inpainted results

5.1.3 RESISC45 dense residential trained Generator Test on RESISC45 medium residential input

In this experiment we analyse a custom MAT generator trained from scratch on the NPWU-RESISC45 dataset's Dense residential class and applying the aforementioned metrics between the inpainted and truth image. The images to be inpainted by the generator also belong in the NPWU-RESISC45 dataset's Medium residential class a class that is similar to the class the generator was trained in.

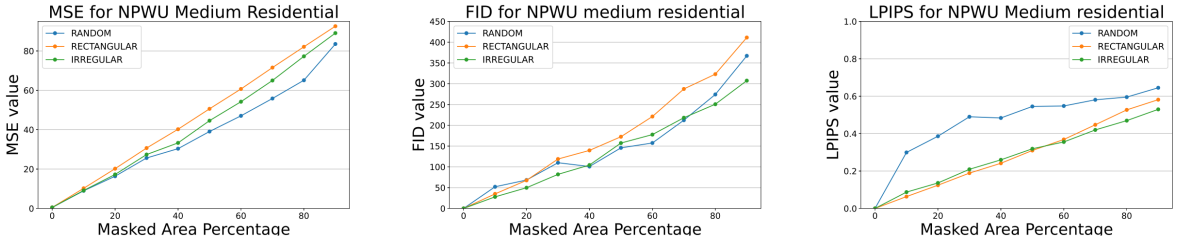


FIGURE 5.6: MSE,FID, LPIPS measurements for generator trained on RESISC45 Medium residential

The MSE metric plot is linear with a clear distinction between the best performing mask that being the random followed by the irregular followed by the rectangular with all of them having a linear rate of increasing MSE, the amplitude of the MSE values is also greater than that of the meadow and Celeba cases. The aforementioned ranking of mask types has been consistent, barring the random bulge phenomenon in Celeba, since during the inpainting the generator has more context for the random and irregular masks compared to the rectangular ones due to these masks being more spread out and leaving some odd ground truth pixels surrounded by masked pixels. While the rectangular mask is a solid patch and the generator is left to fill it with the only context being the surrounding pixels.

The FID plot seems to have a substantially higher amplitude compared to the previous cases with the FID reaching more than 400 while the rectangular masks fare the worst and random and irregular masks oscillate between having the lowest and second lowest FID values. Overall, the FID metric seems to be the most sensitive since it exhibits the greatest amplitude difference when compared to the MSE and LPIPS metrics across all the experiments discussed. Particularly in the final experiment, where the generator was trained on a different class from the ground truth, it is anticipated that the reconstruction metrics would have worse (higher) values.

As for the LPIPS plot it seems again that the rectangular and irregular shaped images have a very similar value and only in for high masked areas can it be said confidently that the irregular masks are better compared to the rectangular masks. The random mask again has the highest LPIPS value a phenomenon consistent across all

LPIPS experiments despite it often having the lowest MSE or FID value. It can be said therefore that the LPIPS metric is especially sensitive random mask reconstruction as it amplifies the LPIPS value of the random mask inpainting having higher value that all the other masks. This is the case because the LPIPS metric measures perceptual similarity and while random masks might offer faithfull reconstructions the reconstructions themselves are blurry and therefore have a high LPIPS value since they are not perceptually consistent. While the other masks' inpainting might not be a reconstruction of the ground truth image the inpainted image at least isn't blurry and offers therefore a higher perceptual consistency thus a higher LPIPS value. This phenomenon is most clearly seen in the Fig. 5.2. The amplitude of the medium residential LPIPS plot lines are also higher than those of the previous experiments. The reason that all three metrics MSE, FID, LPIPS have higher values in this experiment is that in every other case the generator input and training were on the same class while in this experiment the generator input is NPWU medium residential while the generator was trained on NPWU dense residential.

The difference between the NPWU dense residential class and the NPWU medium residential class can be seen especially clearly for large masked areas particularly when applying rectangular masks. As can be seen in Fig 5.7 the random mask inpainting produces a blurry reconstruction of the original image because as was said before there is a lot of context left. This cannot be said for the rectangular image since the solid mask leaves little context for the generator so it ends up filling the mask with what it has been trained on even if there is little coherence between that and the surrounding ground truth. The irregular once again acts as a middle ground for random and rectangular masks leaving some context while also having some solid patches missing.

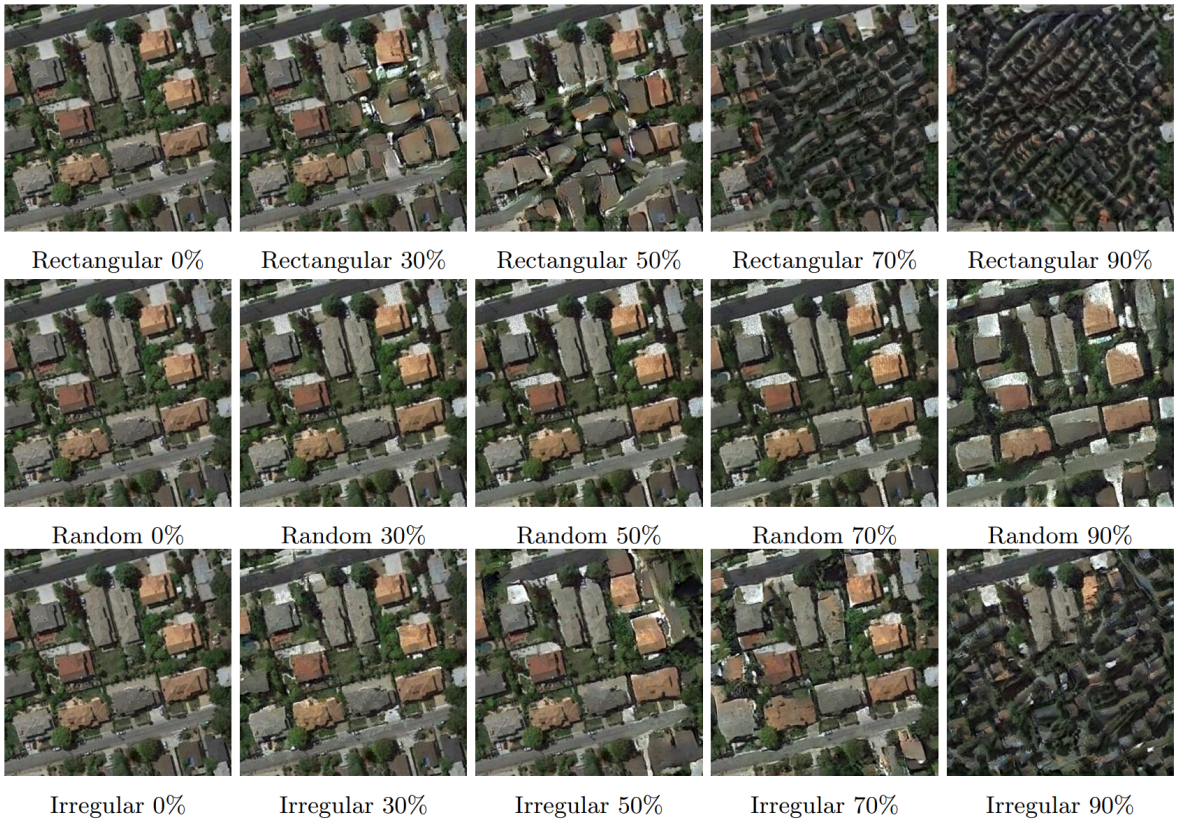
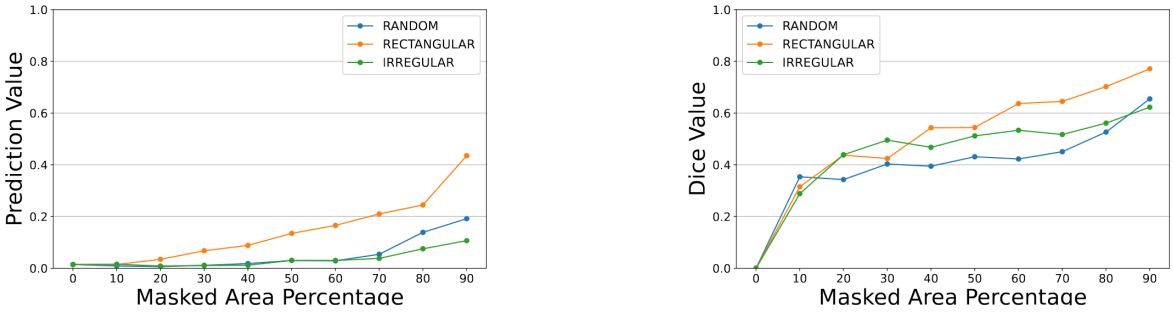


FIGURE 5.7: RESISC45 Medium residential inpainted results

5.2 Generator and Discriminator Testing

This section's experiments aim to test the generator discriminator architecture behaviour on scenarios where the missing regions are inpainted by a generator trained in a class that is different than that of the ground truth image class. We make multiple experiments with generators trained on different classes and with ground truth images of multiple classes of different datasets.

5.2.1 RESISC45 (not meadow) ground truth to RESISC45 (meadow) trained generator



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.8: Prediction and dice values for RESISC45 meadow - not meadow

As evidenced by the out-of-distribution (OOD) prediction value graph, the random and irregular masks exhibit very low values up until 70% coverage, where the rate of change in OOD prediction begins to increase, albeit not significantly. Conversely, the rectangular masks demonstrate a steady increase up to 80% coverage, with a notable bump and exponential increase in OOD prediction at the 90% mark. The final OOD prediction values remain low, not exceeding 45% for the rectangular mask and 20% for the random and irregular masks.

From the dice plot as we can see that overall all the masks are increasing their dice values as the masked area percentage increases with very few instances of decreases whose amplitude decrease is minimal. Generally the rectangular masks offer the best dice value followed irregular and random.

As illustrated by a randomly selected instance of the experiment in Fig. 5.9, the pattern observed in previous experiments is reaffirmed. The rectangular mask creates the largest continuous missing area, and the generator, when reconstructing the missing square, initially attempts to align the outer pixels with the shapes of the adjacent ground truth pixels. However, as the generator continues to reconstruct pixels deeper within the mask, it disregards the surrounding context and fills the

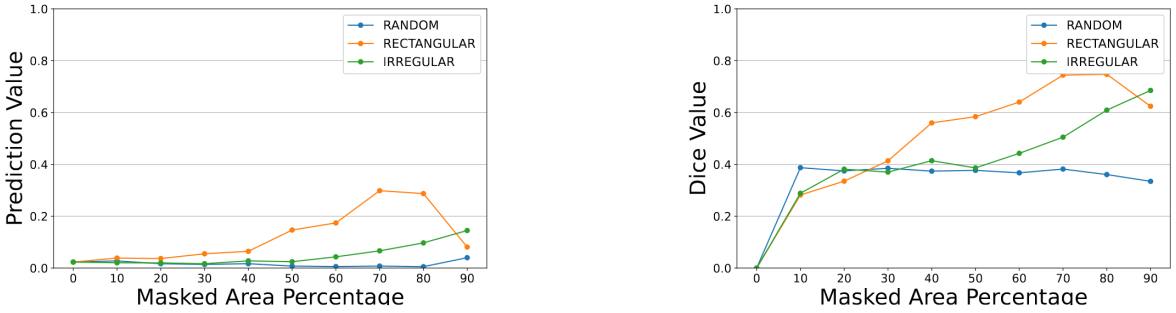
area with solid patches of green pixels, intending to recreate a meadow. This phenomenon is particularly evident in high-coverage rectangular images, such as the 90% coverage example in Fig. 5.9. In this instance, the borders of the missing patch are clearly discernible, with the outer reconstructed pixels exhibiting a pink tint to replicate the adjacent ground truth pixels, while the inner pixels have a green tint, indicating an attempt to reconstruct a meadow with very limited contextual information.

Furthermore, it is observed that for the random masks, the inpainting process, while attempting to reconstruct the ground truth image under high mask coverage, manages to recreate the original shapes. However, the reconstructed pixels exhibit a pink tint instead of the original grey color. This effect is more pronounced in images featuring city landscapes, where the predominant colors are cement and asphalt grey. In contrast, the reconstructions with irregular masks perform even worse than those with random masks. The resulting images appear amorphous, and their out-of-distribution prediction values are classified even lower by the discriminator compared to the reconstructions using random masks.



FIGURE 5.9: RESISC 45 roundabout inpainted results

5.2.2 RESISC45 (not dense-residential) ground truth to RESISC45 (dense-residential) trained generator



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.10: Prediction and dice values for RESISC45 dense residential - not dense residential

The out of distribution prediction value plot is sufficiently different than the one in the NPWU RESISC 45 meadow experiment case. In this plot although the rectangular masks inpainting has a considerably greater value that the other mask's values for 90% masked are percentage a dip can be observed making the irregular mask's inpainting better. Other than that dip the irregular masks has the second highest OOD prediction value followed by the random mask. The OOD prediction values for the rectangular masks doesn't exceed 40% while the irregular and random mask's don't exceed 20% and 10% respectively.

As can be seen from the in distribution dice plot the rectangular mask's inpainting has the highest dice value from 30% mask coverage and up to 80% mask coverage since after that point it's dice value decreases and is overtaken by the irregular masks dice value. The irregular as well as the rectangular masks generally have an increase in dice value as the masked area percentage increases while the random masks remain static and as an extension have a lower dice value than both the irregular and the rectangular masks.

As can be seen from a random instance of the experiment in Fig.5.11 the reason the rectangular masks have a dip at 90% masked area is that for smaller masked areas the inpainted images look like a lower altitude remote sensing image where individual houses can be distinguished while at a certain threshold of masked area the generator inpaints a higher altitude remote sensing image which is harder for the discriminator to classify correctly since very high altitude remote sensing images from different but similar classes tend to converge while the lowest the altitude is the more discrete the differences are and the easier it is for the discriminator to correctly classify. This phenomenon emerges only in the case where the generator is trained on the NPWU RESISC 45 dense residential and it seems to occur mainly in

rectangular masks inpaintings of very high mask coverage it is not so common in irregular mask inpaintings since it emerges when the missing patch is solid enough with limited pockets of ground truth pixels surrounded by the mask. The random masks inpainting as usual reconstructs the image albeit blurry.

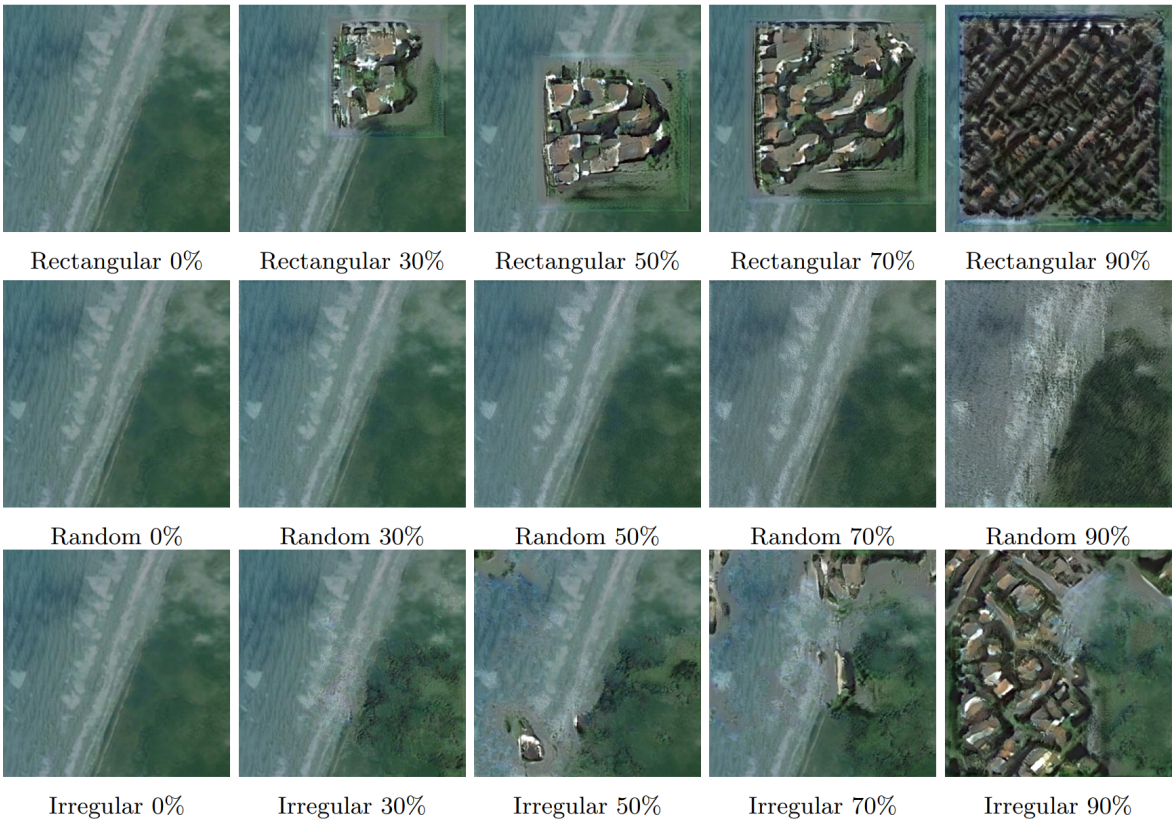
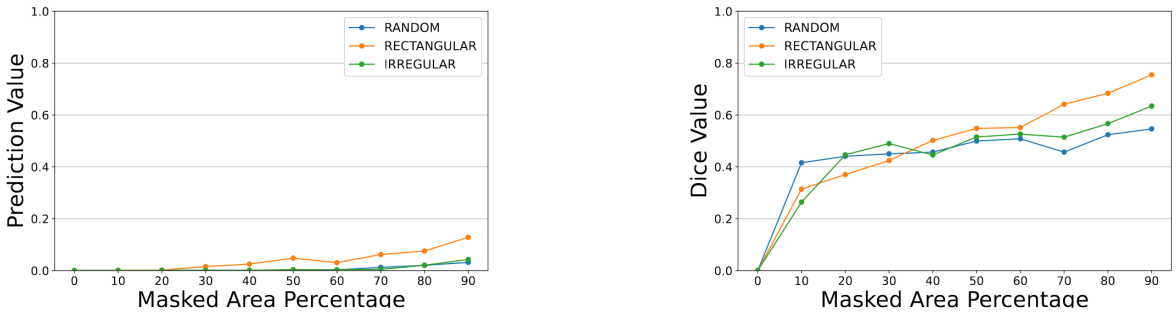


FIGURE 5.11: RESISC 45 beach inpainted results

5.2.3 uc_merced (not meadow) ground truth to RESISC45 (meadow) trained generator



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.12: Prediction and dice values for uc_merced meadow - not meadow

From the out of distribution prediction plot we can see that the rectangular mask's inpainting as per usual has the highest prediction values not going above 20% with the random and the irregular masks tied and under 10%. The rate of increase is extremely slow just as the amplitude of the graph.

As shown in the in-distribution dice plot, all the mask inpaintings demonstrate a steady increase in their dice values as the masked area increases. Among the different mask shapes, the rectangular masks achieve the highest overall dice values, followed by the irregular and random masks.

As illustrated by a random instance of the experiment in Fig. 5.13, the rectangular mask's inpainting reveals a phenomenon observed in previous experiments: the outer pixels of the mask somewhat resemble the surrounding ground truth pixels, while the inner mask pixels form a structure resembling a meadow. When applying random masks, the shapes of the ground truth image are reconstructed; however, as the mask size increases, the reconstructed image becomes increasingly discolored. This discoloration effect is also sometimes observed with irregular masks, where reconstructed pixels may not match the original green or ground truth color. In other cases, the pixels appear green in an attempt to recreate a meadow. An exception occurs at very high irregular mask sizes, resulting in a completely incomprehensible image.



FIGURE 5.13: uc_merced landuse plane inpainted results

5.2.4 uc_merced (not dense residential) ground truth to RESISC45 (dense residential) trained generator

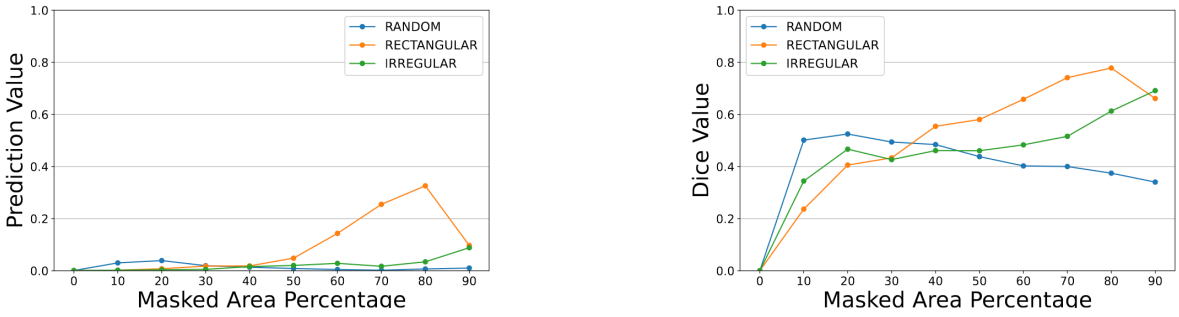


FIGURE 5.14: Prediction and dice values for uc_merced dense residential - not dense residential

It can be seen from the out of distribution prediction graph that the behavior of the inpainting with irregular and random masks is very similar to that of the experiment of section 5.2.3 while the rectangular mask's inpainting is similar to the experiment of section 5.2.2.

As can be seen from the in distribution dice plot the results are very similar to those of the experiment 5.2.2 with the rectangular mask's inpainting having the highest value from one point on and having a dip for very high masked areas and being

overtake by the irregular masks while the random masks is decreasing its dice value. As can be seen from a random instance of the experiment in Fig.5.15 the same phenomenon of higher altitude remote sensing images being inpainted for large masks can be observed explaining the dip in the rectangular mask's inpainting OOD prediction value for large masks. The random masks as always reconstruct the ground truth image albeit discolored and blurry and the irregular masks produce a chaotic result especially when it comes to higher mask sizes.

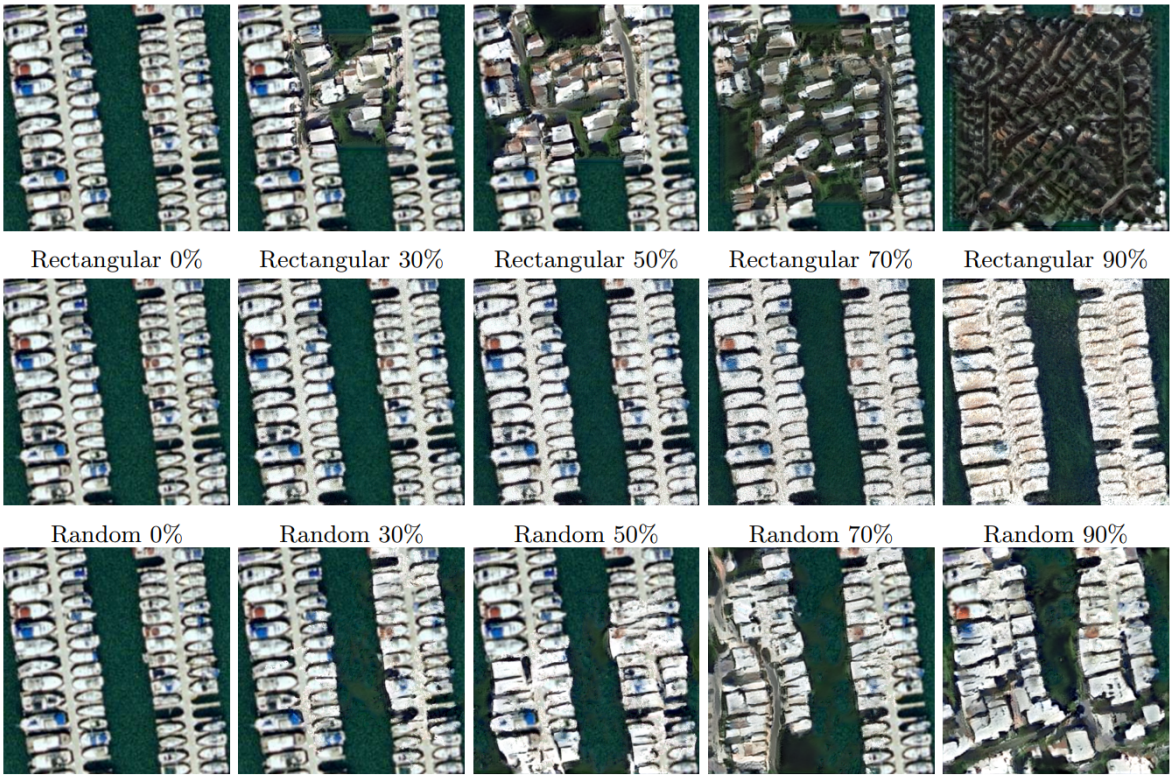


FIGURE 5.15: uc_merced landuse harbor inpainted results

5.3 Generator and Discriminator Testing varying difficulty scenarios

This section's experiments aim to provide different scenarios of varying difficulty to the generator discriminator architecture. Specifically we are employing the generator to inpaint a ground truth image that exhibits varying degrees of similarity to the class of images on which the generator has been trained. These variations include images that are slightly similar, moderately different, and completely different.

5.3.1 Generator trained on RESISC45 meadow input airport

In this experiment we apply a meadow trained generator to airport images. Airport class is highly different compared to the meadow class

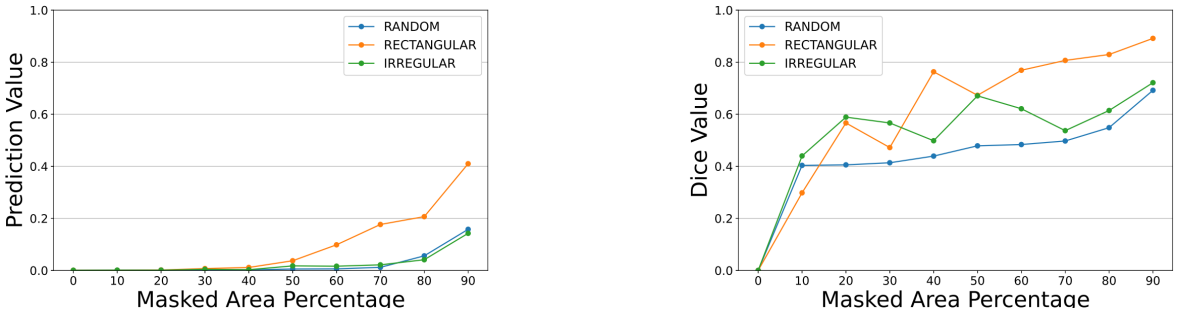


FIGURE 5.16: Prediction and dice values for RESISC45 meadow - airport

As can be seen from the out of distribution prediction value graph the rectangular masks have the highest OOD prediction value with the random and the irregular masks offering the same OOD prediction value. All the masks for small and medium mask patches have an OOD prediction value of near zero while at around 50% mask coverage the OOD prediction values for the rectangular masks start to increase. In very high mask coverage the rectangular masks OOD is just over 40% a big difference from the under 20% OOD value of the other masks.

As observed in the in-distribution dice plot for 0% mask coverage, the value is 0% due to the complete lack of overlap. The inpainting using a rectangular mask consistently outperforms the other two methods across most masked area percentages, exhibiting the highest dice values. The inpainting with an irregular mask generally performs better than the random method but not as well as the rectangular method. While the irregular mask shows fluctuations, it ultimately results in significant improvement. Finally, the inpainting with a random mask demonstrates a steady, gradual increase across the masked area percentages but consistently performs lower than the other masking methods.

A random instance of the experiment is presented in Fig.5.17. The results are consistent with other experiments where the generator was trained on the meadow class. Similar patterns are observed in both experiments: discolored inpainted pixels appear in random and irregular masks, while in rectangular masks, the outer pixels attempt to complement the ground truth pixels, and the inner pixels form a distinct meadow-like structure. Once again, the rectangular masks exhibit the highest OOD prediction values, while the other masks perform less effectively for the same reasons discussed in previous experiments.

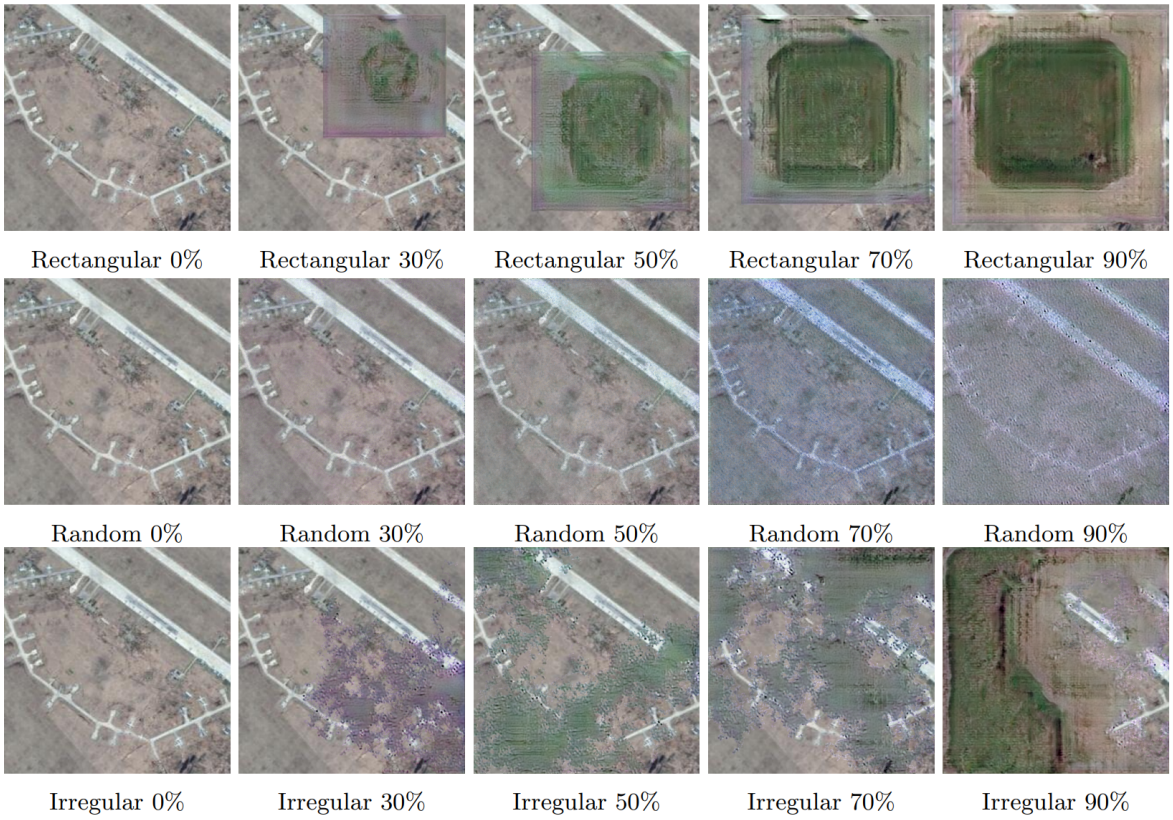
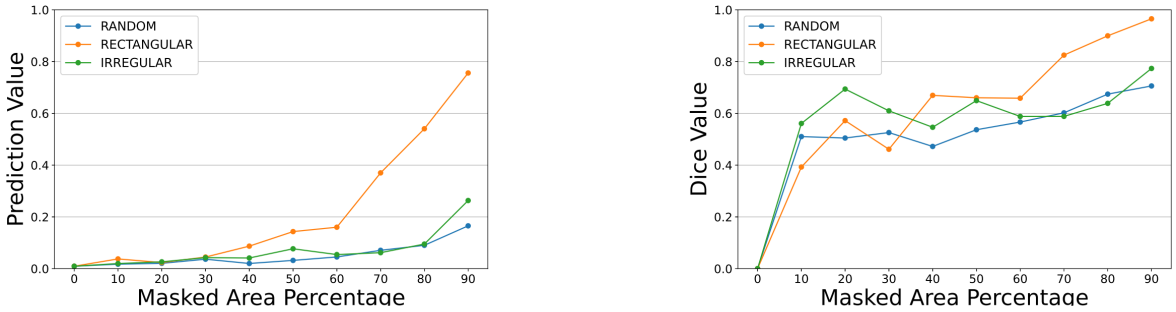


FIGURE 5.17: RESISC45 airport inpainting with meadow trained generator

5.3.2 Generator trained on RESISC45 meadow input river

In this experiment we apply a meadow trained generator to river images. River class is somewhat different compared to the meadow class



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.18: Prediction and dice values for RESISC45 meadow - river

As can be seen from the OOD prediction value graph the rectangular masks have by far the highest OOD prediction value with irregular and random following. Up until 60% the OOD prediction values for all masks are more or less comparable after that point however the rectangular mask's OOD prediction value explodes reaching just below of 80% at the highest masked area percentage while at the same levels of masked area the irregular and random masks are around 20% respectively.

As can be seen from the in distribution dice plot the rectangular mask's inpainting lags behind the other masks for masked areas below 40% while from that point on it has a consistently higher value than the rest of the masks. The irregular mask's reconstruction performs generally better than random masks but not as good as rectangular masks while also showing fluctuations especially in masked areas between 30% and 60%. The random mask's reconstruction as was the case with the previous experiment shows a generally gradual increase across the masked area percentages but has the lowest dice value.

A random instance of the experiment is presented in Fig 5.20. As demonstrated with the rectangular masks, the river itself is inpainted, and due to the abundance of green areas surrounding it, the inpainting result closely resembles a meadow. The primary distinction between the meadow and river classes lies in the presence of the river, as the surrounding areas are not significantly different. Consequently, the inpainting of the river, which is replaced with meadows and occurs almost exclusively with rectangular masks, results in a substantial OOD prediction value as well as a significant ID dice value for the rectangular images. In contrast, these metrics are lower for the other masks, with the OOD prediction value being more affected than the ID dice value.

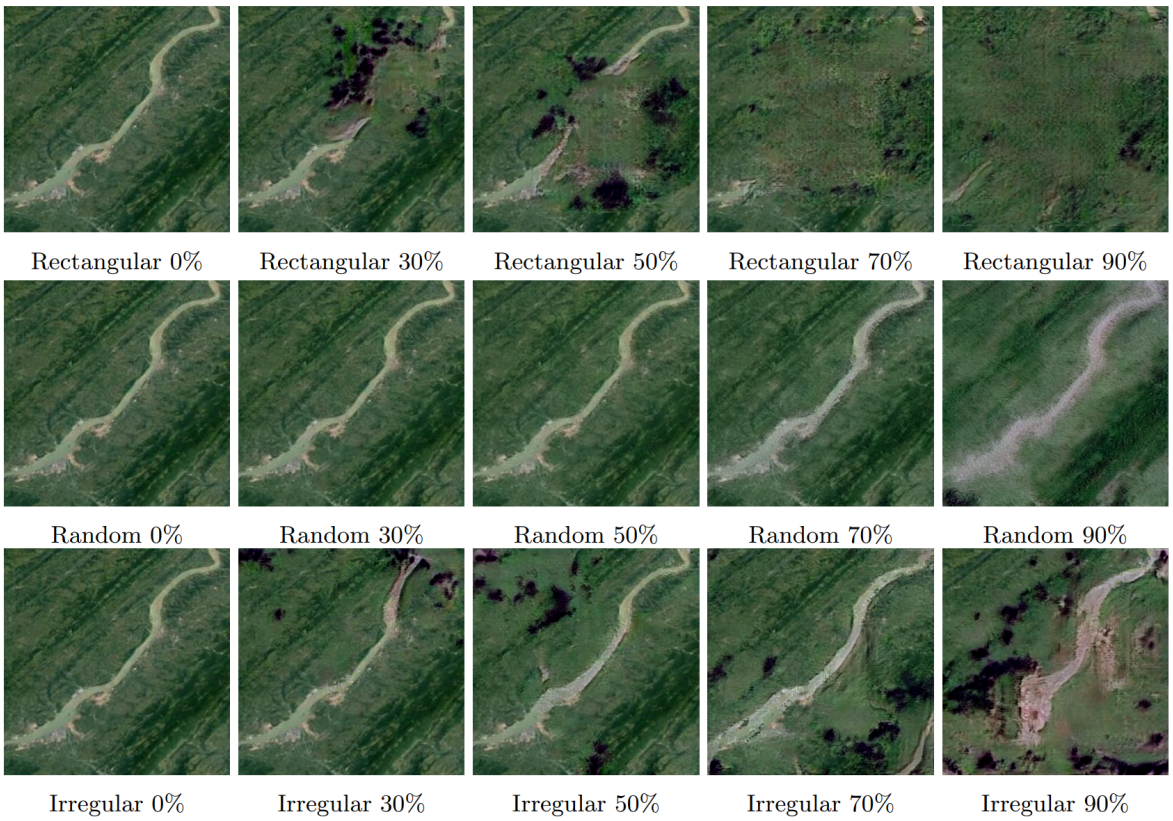
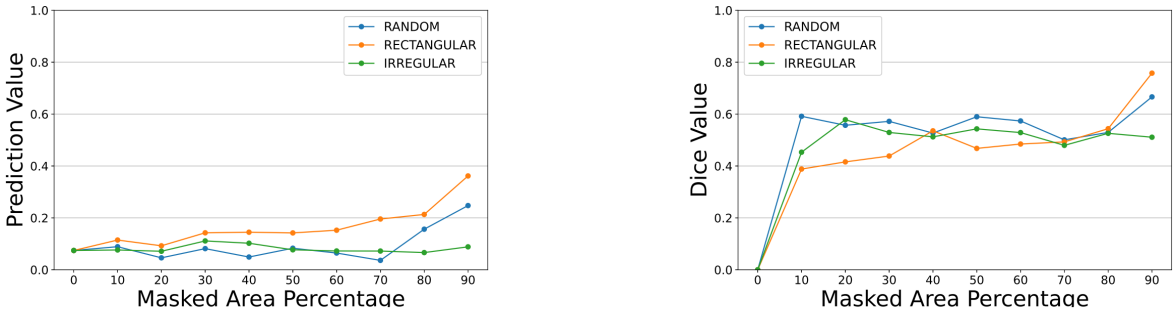


FIGURE 5.19: NPWU RESISC45 river inpainting with meadow trained generator

5.3.3 Generator trained on RESISC45 meadow input forest

In this experiment we apply a meadow trained generator to forest images. Forest class is similar compared to the meadow class.



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.20: Prediction and dice values for RESISC45 meadow - forest

As observed from the OOD prediction value graph, the rectangular masks consistently offer the highest OOD prediction values for all mask sizes, while the irregular masks provide the second-highest OOD prediction values for small to medium-sized masks (up to around 70% mask coverage). Beyond this point, the irregular masks' OOD values decline to become the lowest among the three types. The OOD prediction values for the rectangular masks show a continuous increase, reaching

just below 90% at the largest mask size. In contrast, the random masks exhibit non-monotonic behavior in OOD prediction values up to 70% mask coverage, similar to the irregular masks. However, after 70% mask coverage, the random masks experience a sharp increase, reaching over 20% at their peak. Throughout the experiment, the irregular masks maintain an OOD prediction value of around 10% for all mask sizes, with fluctuations up to 90% mask coverage.

As can be seen from the in distribution dice plot for up to 40% mask coverage the rectangular mask's reconstruction lacks behind the random and irregular mask's reconstruction which fluctuate but have overall very similar dice values. From 40% to 60% there is a clear distinction on the dice values of each mask's reconstruction with random having the highest values followed by irregular and rectangular masks and from 70% on the rectangular mask's reconstruction dice value overtakes the dice value of the rest of the masks.

A random instance of the experiment is presented in Fig 5.21, illustrating the typical patterns produced by the meadow generator. Notably, for irregular and random masks, the inpainted pixels do not exhibit discoloration. Given that the surrounding ground truth pixels are green, the generator inpaints the missing pixels based on its training dataset and the shapes and colors of the surrounding ground truth pixels, resulting in a significant overlap.

Overall, the results indicate that contrary to expectations, the airport images did not exhibit a smaller OOD prediction compared to the river images, nor did the river images have a lower OOD prediction than the forest images. For the rectangular mask in cases with very large masked areas, the OOD prediction values for the random and irregular masks are very similar, despite the significant differences in the ground truth image classes. This suggests that the discriminator determines the OOD value based solely on the masked inpainted area and identifies the surroundings as in-distribution. For smaller masked areas, the forest inpaintings have a higher OOD value, as the context of the ground truth is more similar to that of the meadow class. In the case of river images, once the river is inpainted at around 60% coverage, the OOD prediction value rises rapidly, surpassing the values of the other classes. This is because the inpainted river closely resembles the meadow class, more so than the inpainted forest. As expected, the random and irregular images show that the forest class has the highest OOD prediction value, followed by the river, and finally the airport. However, the differences between them are not significant, especially when comparing the performances of the river and forest classes.

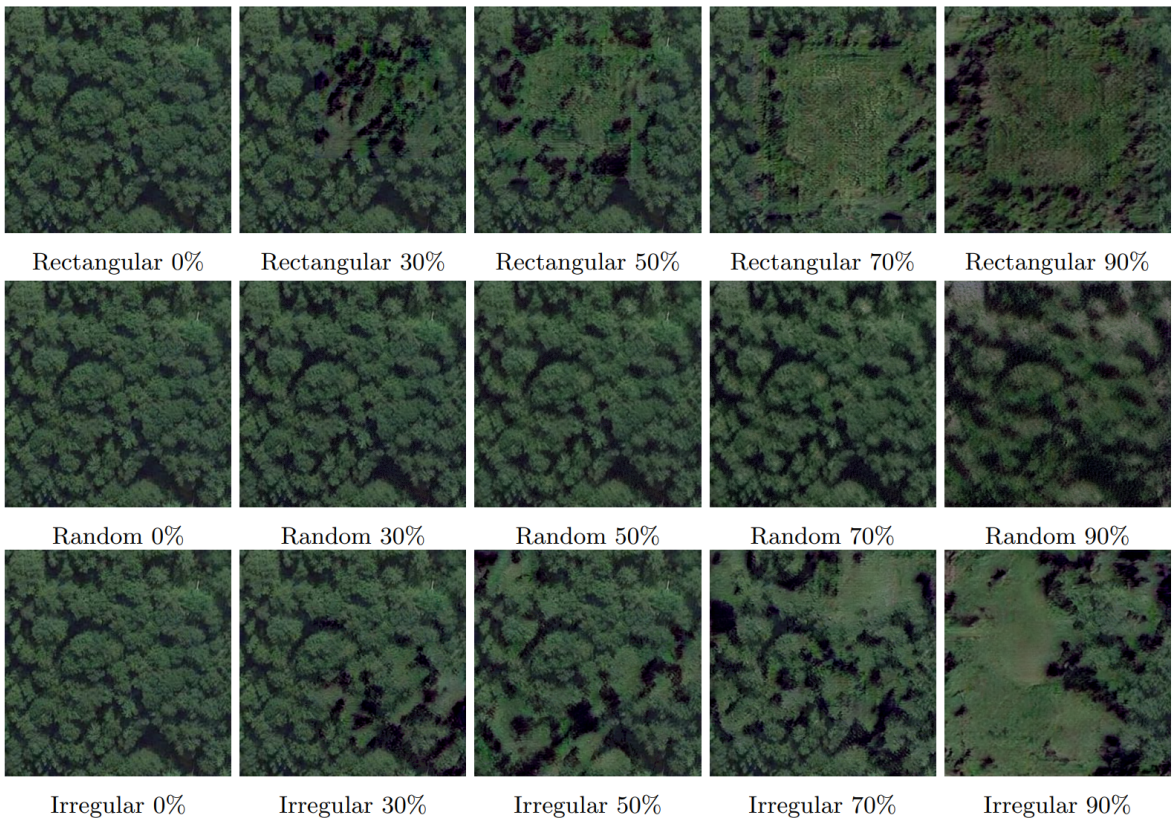
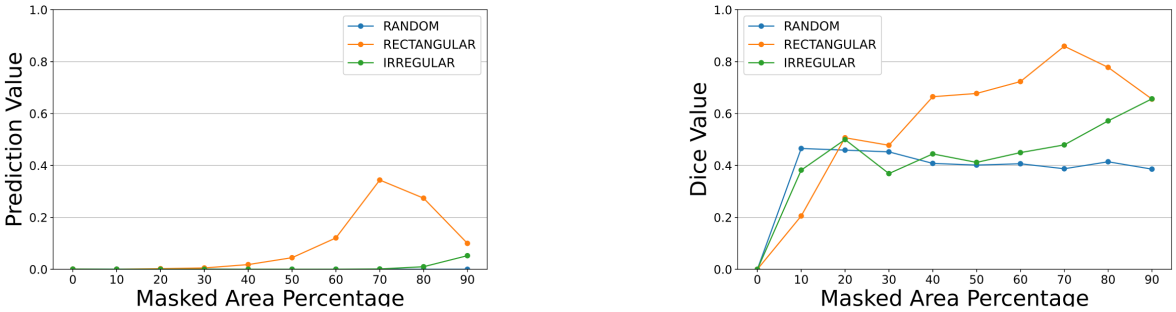


FIGURE 5.21: RESISC45 forest inpainting with meadow trained generator

5.3.4 Generator trained on RESISC45 dense residential input beach

In this experiment we apply a dense residential trained generator to beach images. Beach class is highly different compared to the dense residential class.



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.22: Prediction and dice values for RESISC45 dense residential - beach

As can be seen from the OOD prediction value graph the irregular and random masks remain practically at zero throughout all the mask sizes the rectangular masks remain close to zero up until around 50% mask coverage rapidly increasing their OOD prediction value after that and subsequently falling again after 70% mask coverage forming a bulge.

As can be seen from the in distribution dice plot the rectangular mask's inpainting has a steady increase in it's dice value up to 70% mask coverage where it starts decreasing. The random mask's inpainting decreases steadily while the irregular mask is fluctuating for masked areas up to 50% and then starts increasing. After 50% mask coverage where all the mask types become monotone with the exception of the rectangular masks fall off, the random mask has the lowest dice value the irregular masks have a higher dice value while the rectangular masks have the highest.

As the in distribution dice plot shows the rectangular mask's inpainting dice value is increasing with the masked area and is the highest followed by the irregular mask's inpainting which is also increasing which is in turn followed by the random mask's inpainting which has a steady but slow decline in its dice value. It is also apparent that that the rectangular masks for high masked areas have a drop in their dice values.

As we can see from a random instance of the experiment in Fig.5.23 the rectangular mask's inpainting is the same as as any rectangular masks inpainted with a dense residential trained generator with the characteristic high altitude inpaintings for very large masked sizes and lower altitude inpaintings for smaller sized masks.

It is also worthy to note that the dense residential inpaintings do not have these "transition" pixels that the meadow generators create. It is most clearly seen in the rectangular masks if we compare the rectangular mask's inpaintings with a meadow trained generator as can be seen in Fig.5.17 and with a dense residential trained generator like in Fig.5.23. It is clear that while the outer pixels of the inpainted area try to complement the ground truth pixels in the dense residential trained generator this happens to a much lesser degree and only in small to medium masked image sizes, in larger sizes there is no transition between the ground truth pixels and the inpainted pixels at all. This is probably the case because the dense residential dataset's images naturally contain a lot sharp angles and lines since they are buildings while the meadow dataset's images are the opposite containing more curves and natural shapes. Therefore the generator deems it natural to fill in the masked area with minimal "transition" pixels. It is also apparent from the figure that the random masks' inpainting result in big masked areas is not just a reconstruction as it was in previous cases but for high masked areas the inpainted image constitutes of a damaged or imperfectly reconstructed ground truth image. Finally the irregular masks once again provide a middle ground between the random and rectangular masks with their inpainting being similar to either depending on the density of the pixels in that area.

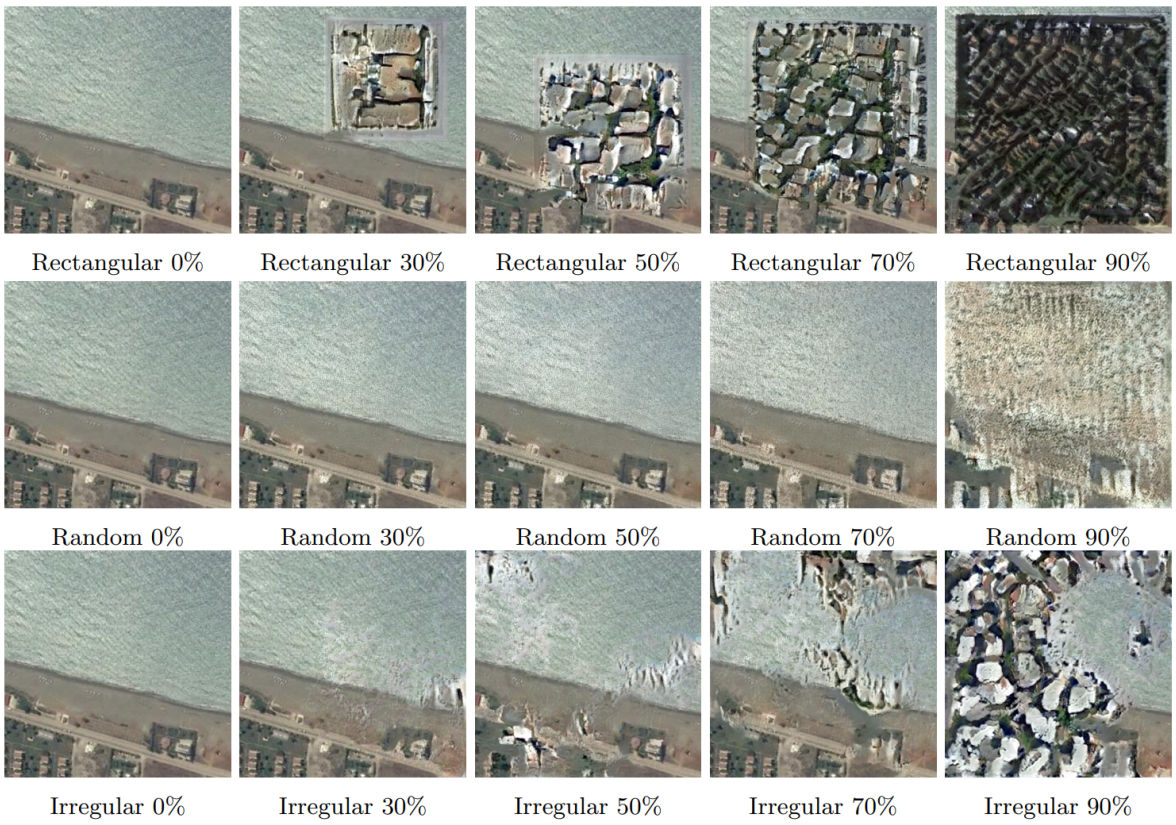
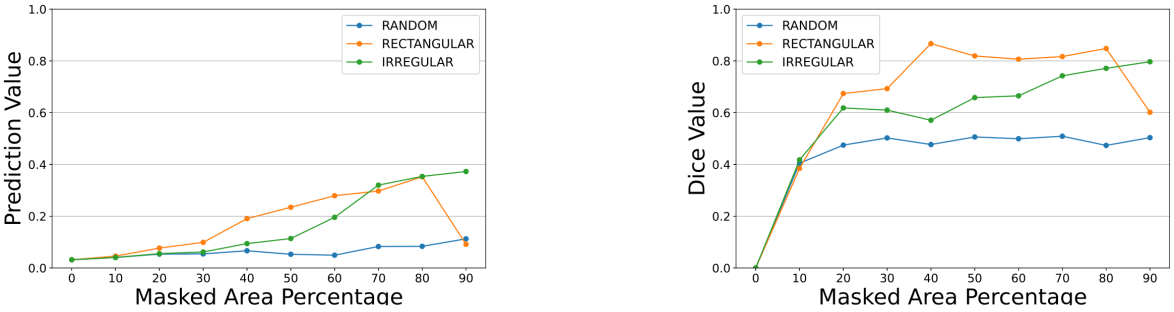


FIGURE 5.23: RESISC45 beach inpainting with dense residential trained generator

5.3.5 Generator trained on RESISC45 dense residential input commercial

In this experiment we apply a dense residential trained generator to commercial (area) images. Commercial class is somewhat different compared to the dense residential class.



(a) Out of distribution prediction value

(b) In distribution dice value

FIGURE 5.24: Prediction and dice values for RESISC45 dense residential - commercial

As observed in the OOD prediction value graph, for mask coverage up to 60%, the rectangular masks yield the highest OOD prediction values, followed by the irregular masks and then the random masks. At 70% mask coverage, the irregular masks overtake the rectangular masks in OOD prediction value. At 80% mask coverage, the OOD prediction value for rectangular masks drops to around 10%, and they are overtaken by the random masks, which also hover around 10%. Meanwhile, the irregular masks maintain an OOD prediction value just below 40%.

As can be seen from the in distribution dice plot the rectangular mask's inpainting has the highest dice value with the exception of the value for 90% mask coverage where it falls off. The second lowest value is that of the irregular masks which overtakes the rectangular masks for 90% masked are and followed by the random masks which seem to have a steady value for all masked areas except 0%.

A random instance of the experiment can be seen in Fig.5.25. The results of the inpainting are largely the same with those of the experiment in section 5.3.4. The main difference between those two inpainting is that in this inpainting where the ground truth image class is closer to that of the previous experiment since they are both cityscapes. Although random masks can reconstruct the overall shapes of buildings, especially in highly masked areas, they fail to capture details on the roofs. Consequently, the final image depicts the building shapes with overly smooth and continuous roofs. This issue is evident in Fig. 5.25, which shows an example with 90% random mask coverage.



FIGURE 5.25: RESISC45 commercial inpainting with dense residential trained generator

5.3.6 Generator trained on RESISC45 dense residential input medium residential

In this experiment we apply a dense residential trained generator to medium residential images. Medium residential class is similar compared to the dense residential class.

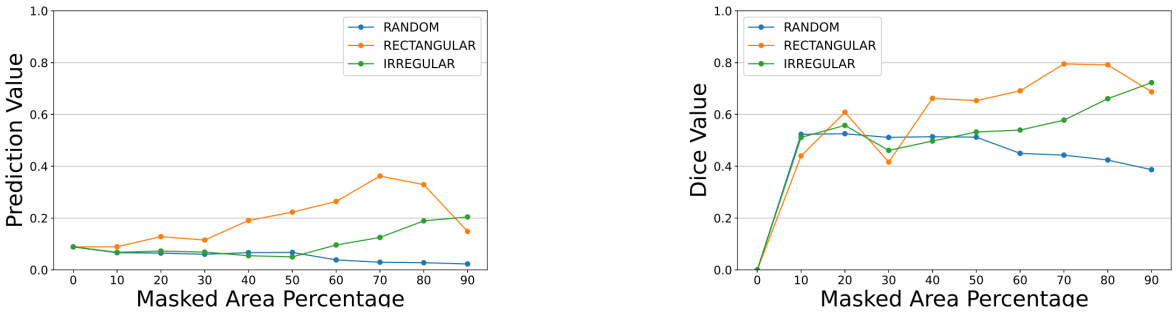


FIGURE 5.26: Prediction and dice values for RESISC45 dense residential - medium residential

As depicted in the OOD prediction value graph, up to 50% mask coverage, the OOD prediction values for the irregular and random masks are nearly identical, maintaining a steady level of around 10%, whereas the rectangular masks show a consistent

increase. Beyond 60% mask coverage, the irregular masks begin to surpass the random masks, showing a steady increase, while the OOD prediction values for the random masks steadily decrease as the masked area increases. By 70%, the OOD prediction values for the rectangular masks start to decline, and by 90%, they drop significantly, being surpassed by the irregular masks which exhibit a steeper decline.

As observed in the in-distribution dice plot, the inpainting results for the random masks show a consistently decreasing dice value, whereas the irregular masks exhibit a generally increasing trend with values surpassing those of the random masks. The rectangular masks, however, display a sudden drop at both 30% and 90% mask coverage in this experiment. Throughout these experiments, the ID Dice metric shows greater stability compared to the OOD prediction value. The OOD prediction value remains consistently low, barely above zero for most mask sizes. In contrast, the ID Dice value demonstrates more pronounced changes in response to variations in mask size and differences in the ground truth dataset.

A random instance of the experiment is depicted in Fig. 5.27. The results align closely with those obtained from the experiment discussed in Section 5.3.5, particularly regarding the behavior of random masks' inpainting. Random masks reconstruct larger structures but often miss finer details. This characteristic explains why the inpainting performance of random masks, as measured by both Dice scores and OOD prediction values, consistently underperforms when using a generator trained on dense residential areas compared to irregular masks. Interestingly, this discrepancy is not observed when using a generator trained on meadow areas.

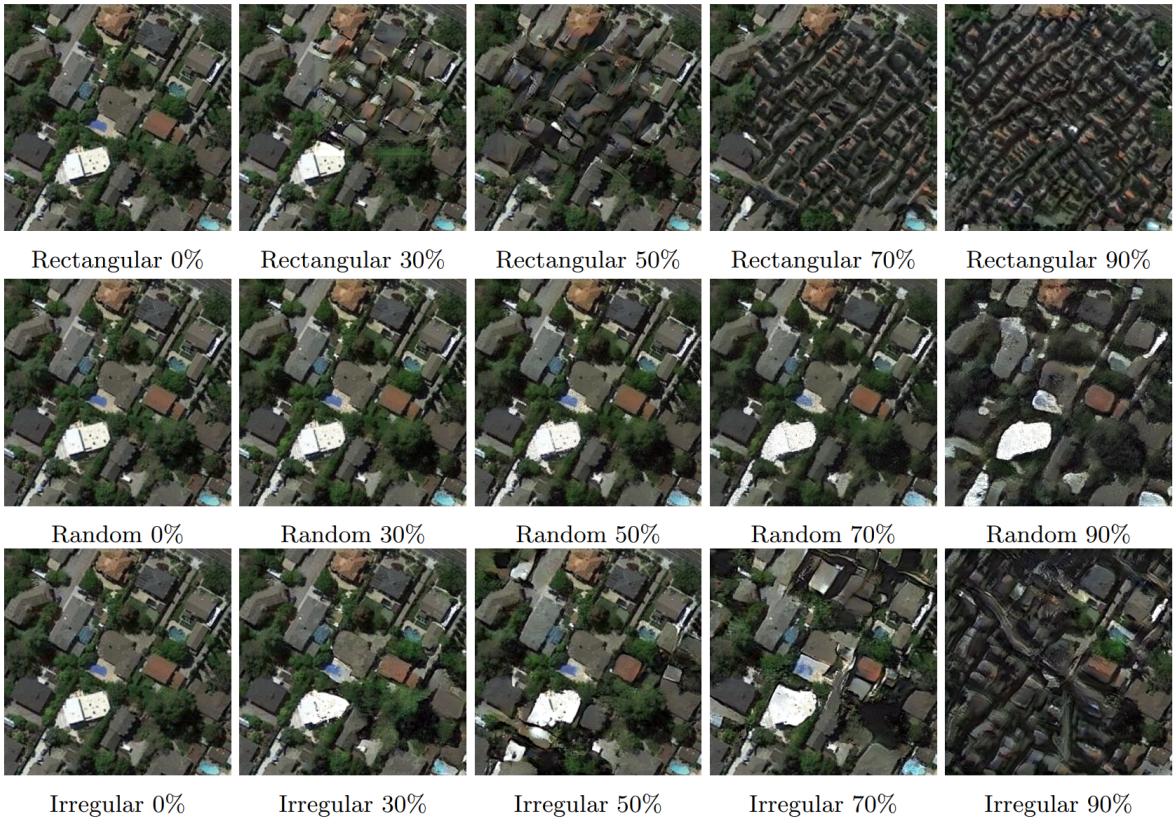


FIGURE 5.27: RESISC45 medium inpainting with dense residential trained generator

Chapter 6

Conclusion

This study conducted an in-depth analysis of hallucination detection in an image inpainting environment across multiple scenarios. It involved evaluating metrics for various mask types and sizes using images from two distinct datasets. The results were analyzed utilizing a two-class discriminator and a Grad-CAM model.

This study revealed that besides the size the mask type significantly affects both the inpainted result and the hallucination metrics. Rectangular masks consistently achieved the highest ID Dice and OOD prediction values but faced challenges at very high mask coverage for specific classes. Irregular masks performed variably, generally outperforming random masks but not matching the effectiveness of rectangular masks. Random masks showed steady improvement with increased mask coverage but consistently performed less effectively compared to other types.

It has also been apparent that each class trained generator has different inpainting pattern and in some cases a generator can behave differently according to mask size this can be seen especially in the case of the dense residential trained generator which on large masks changes the altitude of the inpainted image while the inpainted region itself regardless of mask size has a minimal transition border region.

Finally it has been established the best metrics for hallucination detection are the In distribution (ID) dice metric accounting for in distribution (ID) pixels and the out of distribution (OOD) prediction value. Furthermore the ID dice metric is more robust though not as sensitive compared to the OOD prediction value. While the FID is overall the best reconstruction metric.

Chapter 7

Future Work

This work explored the phenomenon of hallucinations on image inpainting systems and made head ways into the problem of quantifying them. However, there is still room for improvement and further research

7.1 Generators

While we utilized the MAT architecture for the generator to achieve more accurate results, various types of generators could be employed in each experiment. By averaging the results across different generators, a more comprehensive understanding could be attained, thereby approximating real-world applications more closely. Moreover, further training of the generator could be explored to enhance performance.

7.2 Datasets

For the remote sensing component of our study, we utilized only the NPWU RESISC45 and the UC Merced Land Use datasets. A more comprehensive study could be achieved by incorporating a greater number of datasets. Additionally, our findings indicate that each class can exhibit different results. Therefore, a study employing a broader range of classes beyond the NPWU RESISC45 meadow and dense residential for generator training and testing would yield a more comprehensive result.

7.3 Discriminators

In this work a two class classifier was used to approach a one class classification (OCC). A subsequent study therefore could make use of different types of OCC classifiers so as to achieve more comprehensive results.

7.4 Mask types and sizes

In this study, we have demonstrated that different mask types produce significantly varied inpainting results. However, our investigation was limited to only three types of masks. Future research could expand the range of masks used, with masks designed to mimic cloud coverage or line masks simulating line noise caused by a satellite's faulty readout electronics, with even finer mask coverage step size than the 10% used in this study. Researching the effects of such mask types could be beneficial for real-world applications, as it would enable the development of models capable of quantifying hallucinations in inpainted damaged real-world images.

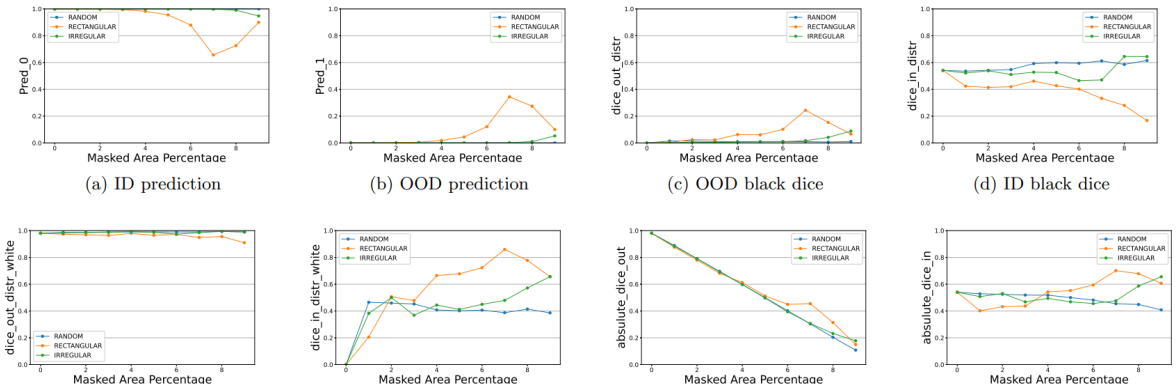
Appendix A

Metrics

In this appendix, we present the comprehensive metrics obtained from several experiments. A detailed breakdown of these metrics is provided below. Please refer to Fig. 4.7 for a visual aid.

- The in-distribution (ID) and out-of-distribution (OOD) prediction values indicate the confidence that the discriminator assigns to the inpainted image being classified as either the ID class (ground truth class) or the OOD class (inpainted class), respectively.
- The ID and OOD black Dice metrics represent the Sørensen–Dice coefficient, accounting solely for the black pixels between the mask and the thresholded OOD heatmap in the OOD case, and between the inverted heatmap and the thresholded ID heatmap in the ID case.
- The ID and OOD white Dice metrics represent the Sørensen–Dice coefficient, accounting solely for the white pixels between the mask and the thresholded OOD heatmap in the OOD case, and between the inverted heatmap and the thresholded ID heatmap in the ID case.
- The ID and OOD black and white Dice metrics represent the Sørensen–Dice coefficient for all pixels between the mask and the thresholded OOD heatmap in the OOD case, and between the inverted heatmap and the thresholded ID heatmap in the ID case.

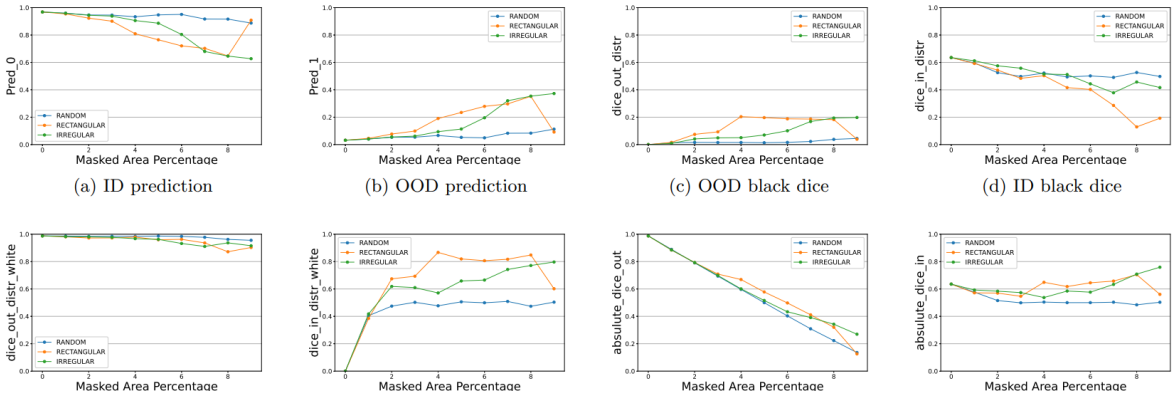
As is evident by the results below the only metrics that are the most closely matched with the y=x line representing the increasing rate of hallucinations with increasing mask coverage, was the Dice metric between the thresholded in-distribution (ID) heatmap and the inverted mask as well as the OOD prediction value.



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

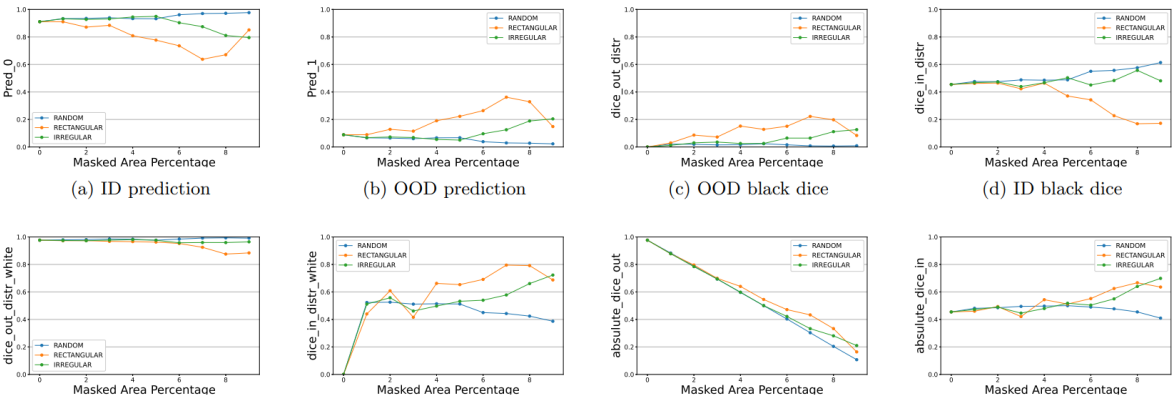
FIGURE A.1: metrics for section 5.3.1



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

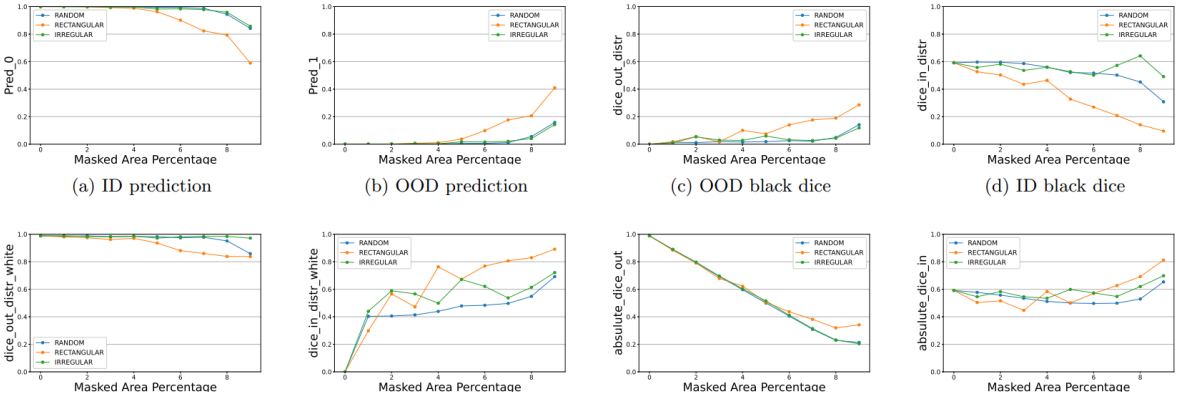
FIGURE A.2: metrics for section 5.3.2



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

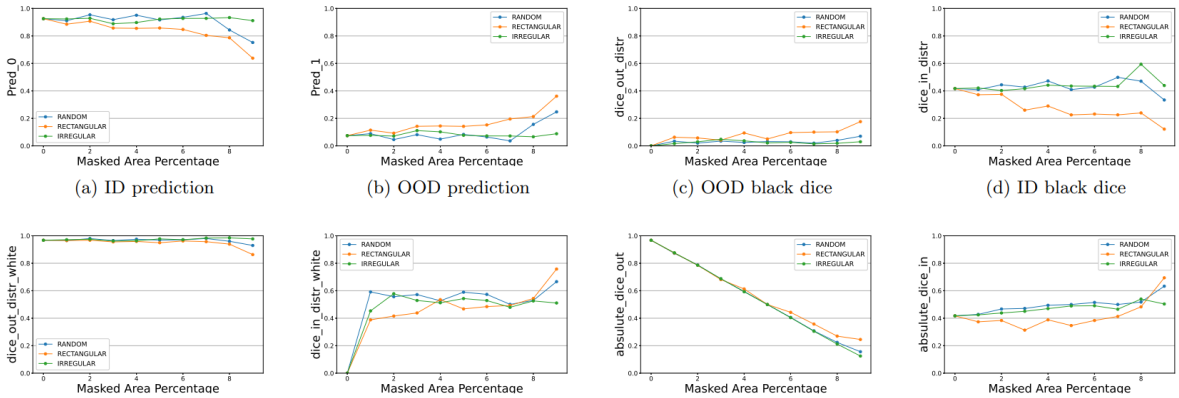
FIGURE A.3: metrics for section 5.3.3



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

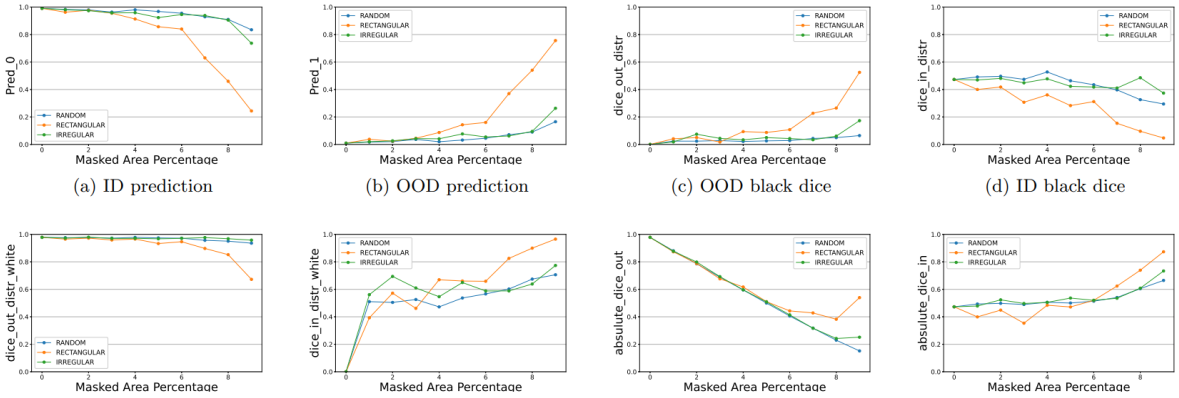
FIGURE A.4: metrics for section 5.3.4



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

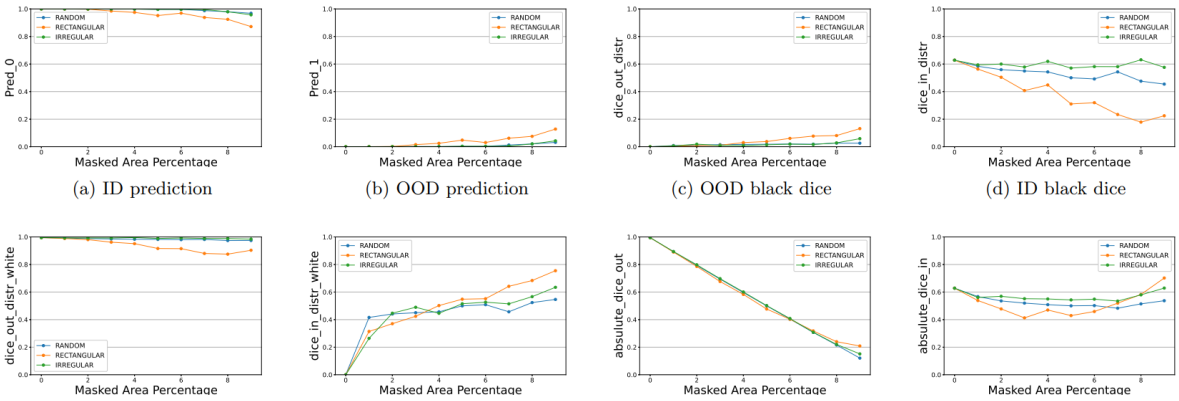
FIGURE A.5: metrics for section 5.3.5



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

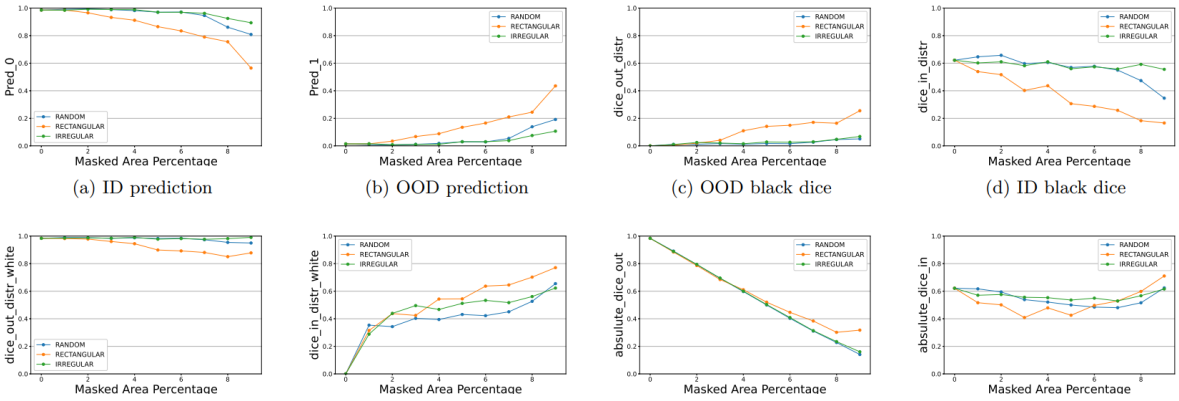
FIGURE A.6: metrics for section 5.3.6



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

FIGURE A.7: metrics for section 5.2.1



(a) ID prediction (b) OOD prediction (c) OOD black dice (d) ID black dice

(e) OOD white dice (f) ID white dice (g) OOD black and white dice (h) ID black and white dice

FIGURE A.8: metrics for section 5.2.3

Appendix B

Generator Training Instances

In this appendix, we present examples from the first and last epochs of each trained generator, along with their corresponding ground truth images.

B.1 Meadow trained generator



FIGURE B.1: Ground Truth images

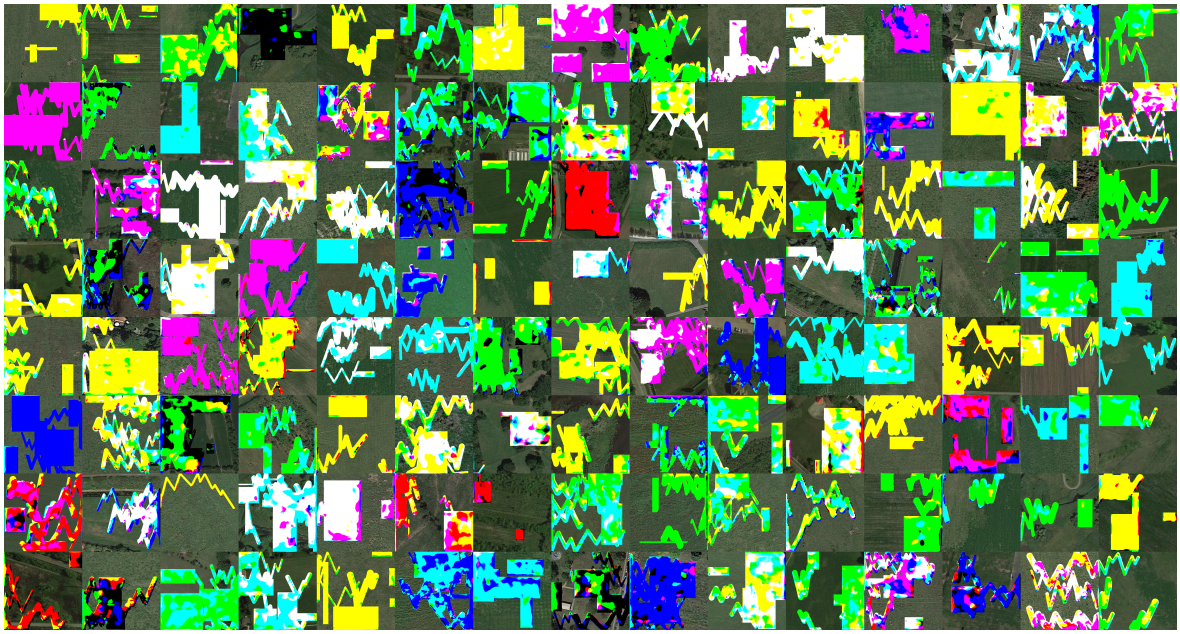


FIGURE B.2: 0 epochs inpainted results



FIGURE B.3: 160 epochs inpainted results

B.2 Dense residential trained generator

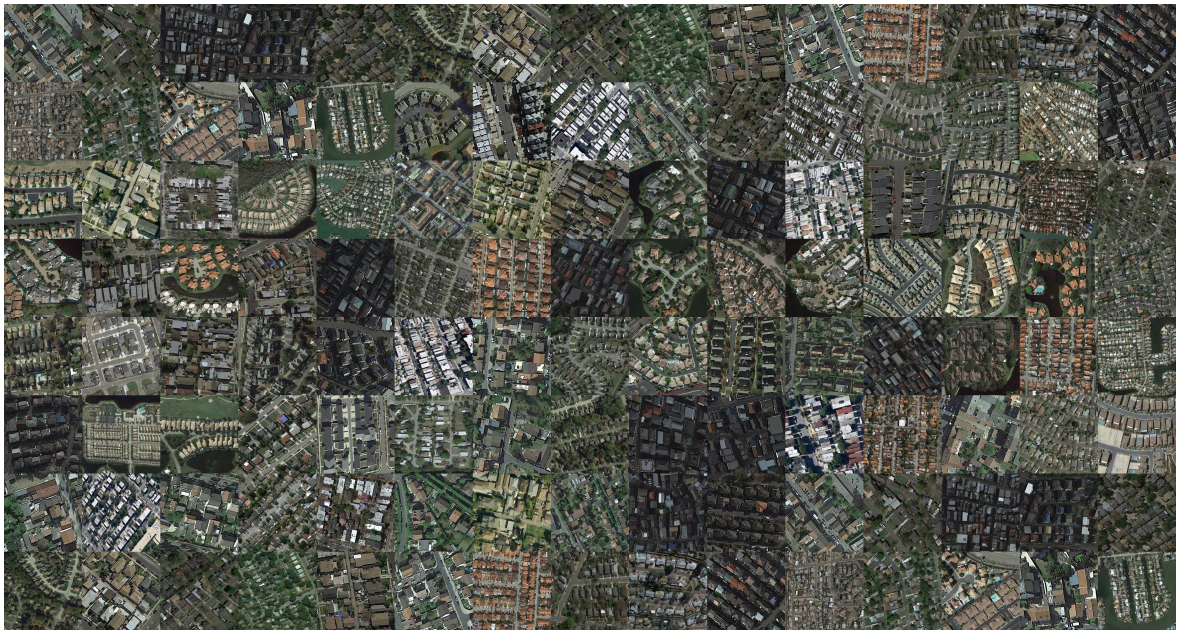


FIGURE B.4: Ground Truth images

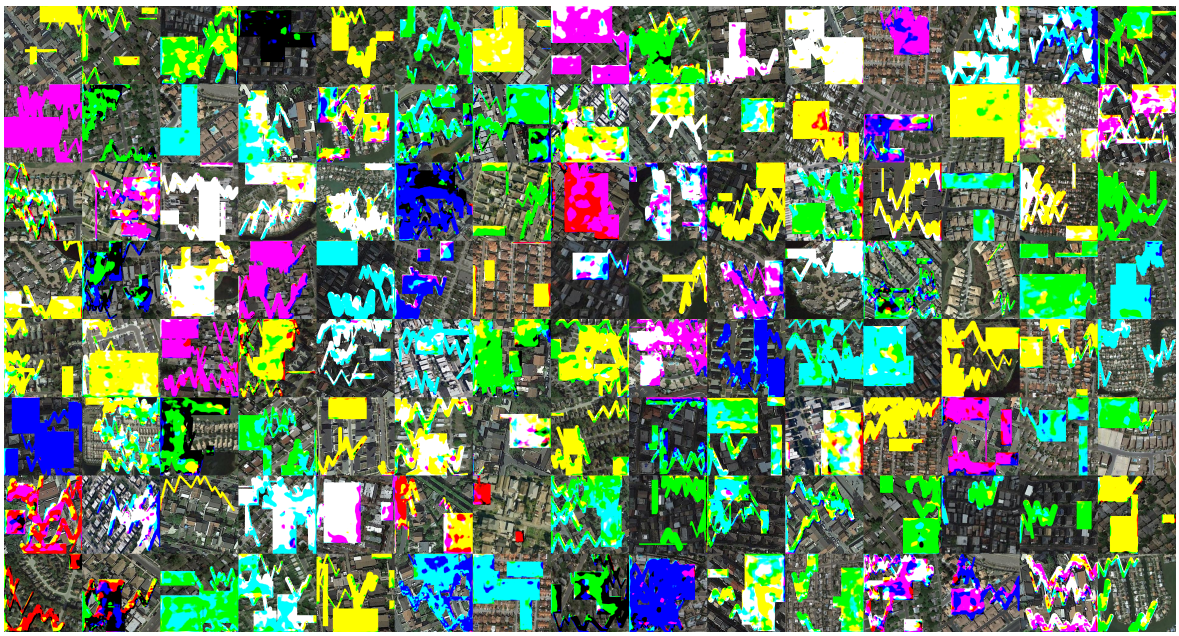


FIGURE B.5: 0 epochs inpainted results

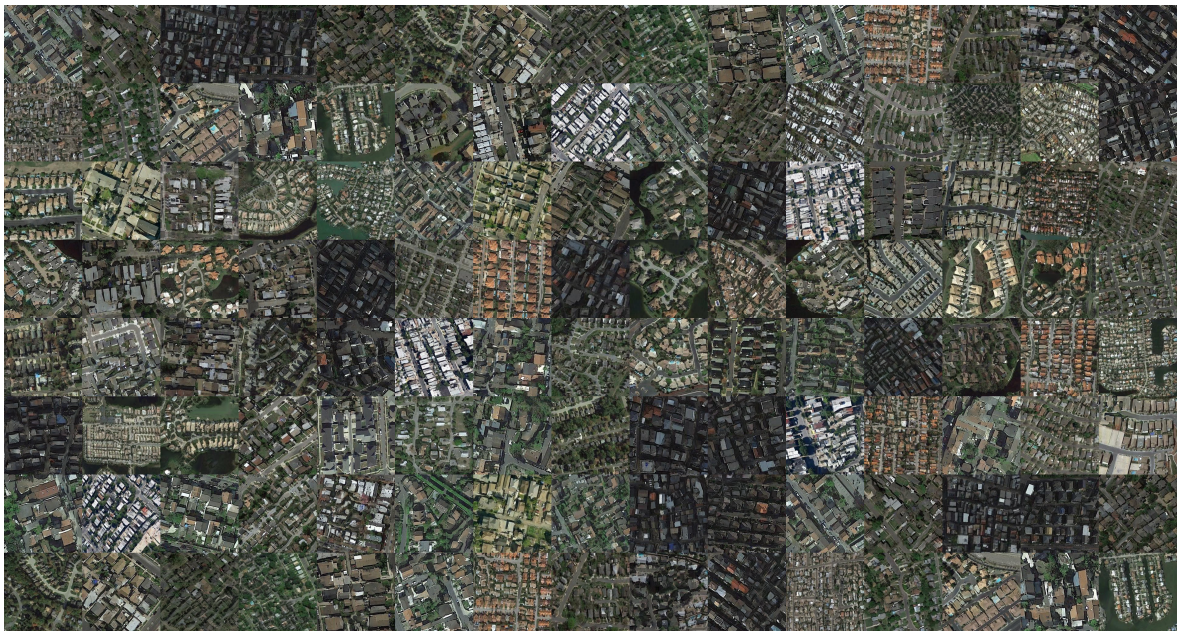


FIGURE B.6: 304 epochs inpainted results

References

[1] Zhen Qin et al. "Image inpainting based on deep learning: A review". In: *Displays* 69 (2021), p. 102028. ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2021.102028. URL: https://www.sciencedirect.com/science/article/pii/S0141938221000391.

[2] Alexander Pondaven et al. "Convolutional Neural Processes for Inpainting Satellite Images". In: *ArXiv* abs/2205.12407 (2022). URL: https://api.semanticscholar.org/CorpusID:249062784.

[3] Advait Kumar et al. "RSINet: Inpainting Remotely Sensed Images Using Triple GAN Framework". In: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium* (2022), pp. 143–146. URL: https://api.semanticscholar.org/CorpusID:246822499.

[4] Wenli Huang et al. "Image Inpainting with Bilateral Convolution". In: *Remote Sensing* 14.23 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14236140. URL: https://www.mdpi.com/2072-4292/14/23/6140.

[5] Dimitrios A. Kasampalis et al. "Contribution of Remote Sensing on Crop Models: A Review". In: *Journal of Imaging* 4.4 (2018). ISSN: 2313-433X. DOI: 10.3390/jimaging4040052. URL: https://www.mdpi.com/2313-433X/4/4/52.

[6] Debra F. Laefer. "Harnessing Remote Sensing for Civil Engineering: Then, Now, and Tomorrow". In: *Applications of Geomatics in Civil Engineering*. Ed. by Jayanta Kumar Ghosh and Irineu da Silva. Singapore: Springer Singapore, 2020, pp. 3–30. ISBN: 978-981-13-7067-0.

[7] Hojat Shirmard et al. "A review of machine learning in processing remote sensing data for mineral exploration". In: *Remote Sensing of Environment* 268 (2022), p. 112750. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2021.112750. URL: https://www.sciencedirect.com/science/article/pii/S0034425721004703.

[8] Ethem Alpaydin. *Machine learning*. MIT press, 2021.

[9] L. D. Harmon. "Artificial Neuron". In: *Science* 129.3354 (1959), pp. 962–963. DOI: 10.1126/science.129.3354.962. eprint: https://www.science.org/doi/pdf/10.1126/science.129.3354.962. URL: https://www.science.org/doi/abs/10.1126/science.129.3354.962.

[10] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks". In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[11] "Activation Function medium". In: URL: https://medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e.

[12] "Loss functions towardsdatascience". In: URL: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9.

[13] "Categorical Cross Entropy shiksha". In: URL: https://www.shiksha.com/online-courses/articles/cross-entropy-loss-function/.

[14] "back propagation neural networks and deep learning". In: URL: http://neuralnetworksanddeeplearning.com/chap2.html?ref=jeremyjordan.me.

[15] "backpropagation nvidia". In: URL: https://developer.nvidia.com/blog/deep-learning-nutshell-history-training/#training/.

[16] backpropagation jeremyjordan. "https://www.jeremyjordan.me/neural-networks-training/". In.

[17] "Optimizers towardsdatascience". In: URL: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[18] "Optimizers medium". In: URL: https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0.

[19] "Optimizers medium". In: URL: https://medium.com/analytics-vidhya/this-blog-post-aims-at-explaining-the-behavior-of-different-algorithms-for-optimizing-gradient-46159a97a8c1.

[20] "Overfitting IBM". In: URL: https://www.ibm.com/topics/overfitting.

[21] "Overfitting AWS". In: URL: https://aws.amazon.com/what-is/overfitting/.

[22] "Underfitting AWS". In: URL: https://www.ibm.com/topics/underfitting/.

[23] "Overfitting and Underfitting linkedin". In: URL: https://www.linkedin.com/pulse/regularization-jaime-arturo-blanco-gutierrez?trk=pulse-article_more-articles_related-content-card.

[24] "CNNs IBM". In: URL: https://www.ibm.com/topics/convolutional-neural-networks/.

[25] "CNNs towardsdatascience". In: URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939/.

[26] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458.

[27] "CNNs saturncloud". In: URL: https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way//.

[28] "Pooling researchgate". In: URL: https://www.researchgate.net/figure/Pooling-layer-operation-oproaches-1-Pooling-layers-For-the-function-of-decreasing-the_fig4_340812216.

[29] "Global Average Pooling medium". In: URL: https://medium.com/aaweg-interview/computer-vision-global-average-pooling-d2bbdfbd713f.

[30] "Nvidia Transformers". In: URL: https://blogs.nvidia.com/blog/what-is-a-transformer-model/.

[31] "IBM Transformers". In: URL: https://www.ibm.com/topics/transformer-model.

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

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

[34] Efros - A.A. and Leung - T.K. "Texture synthesis by non-parametric sampling". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 12. 1999, pp. 1033–1038. DOI: 10.1109/ICCV.1999.790383.

[35] Olivier Le Meur and Christine Guillemot. "Super-Resolution-Based Inpainting". In: vol. 7577. Oct. 2012. ISBN: 978-3-642-33782-6. DOI: 10.1007/978-3-642-33783-3_40.

[36] Marcelo Bertalmío et al. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000). URL: https://api.semanticscholar.org/CorpusID:308278.

[37] Connelly Barnes et al. "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing". In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (Aug. 2009).

[38] C. Ballester et al. "Filling-in by joint interpolation of vector fields and gray levels". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 10 (Feb. 2001), pp. 1200–11. DOI: 10.1109/83.935036.

[39] Marcelo Bertalmio et al. "Image inpainting". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424. ISBN: 1581132085. DOI: 10.1145/344779.344972. URL: https://doi.org/10.1145/344779.344972.

[40] Norihiko Kawai, Tomokazu Sato, and Naokazu Yokoya. "Image Inpainting Considering Brightness Change and Spatial Locality of Textures". In: *International Conference on Computer Vision Theory and Applications*. 2009. URL: https://api.semanticscholar.org/CorpusID:12933462.

[41] Lei He et al. "An Adaptive Image Inpainting Method Based on Continued Fractions Interpolation". In: *Discrete Dynamics in Nature and Society* 2018 (June 2018), pp. 1–16. DOI: 10.1155/2018/9801361.

[42] Hongying Zhang and Shimei Dai. "Image Inpainting Based on Wavelet Decomposition". In: *Procedia Engineering* 29 (Dec. 2012), pp. 3674–3678. DOI: 10.1016/j.proeng.2012.01.551.

[43] Haixia Wang et al. “Exemplar-based Image Inpainting Using Structure Consistent Patch Matching”. In: *Neurocomputing* 269 (May 2017). DOI: 10.1016/j.neucom.2016.08.149.

[44] Viren Jain and Hyunjune Seung. “Natural Image Denoising with Convolutional Networks.” In: Jan. 2008, pp. 769–776.

[45] Junyuan Xie, Linli Xu, and Enhong Chen. “Image Denoising and Inpainting with Deep Neural Networks”. In: *Advances in Neural Information Processing Systems* 1 (Jan. 2012).

[46] Deepak Pathak et al. “Context Encoders: Feature Learning by Inpainting”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2536–2544. DOI: 10.1109/CVPR.2016.278.

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

[48] Lei Zhao et al. “UCTGAN: Diverse Image Inpainting Based on Unsupervised Cross-Space Translation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[49] Suresh Babu Golla Sunitha Abburu. “Satellite Image Classification Methods and Techniques: A Review”. In: *International Journal of Computer Applications* 119.8 (June 2015), pp. 20–25. ISSN: 0975-8887. DOI: 10.5120/21088-3779. URL: https://ijcaonline.org/archives/volume119/number8/21088-3779/.

[50] Geoffrey H. Ball and David J. Hall. “ISODATA, A NOVEL METHOD OF DATA ANALYSIS AND PATTERN CLASSIFICATION”. In: 1965. URL: https://api.semanticscholar.org/CorpusID:53887616.

[51] Vikramaditya R. Jakkula. “Tutorial on Support Vector Machine (SVM)”. In: 2011. URL: https://api.semanticscholar.org/CorpusID:15115403.

[52] Norah Alrebdi et al. “Reinforcement Learning in Image Classification: A Review”. In: Jan. 2022, pp. 79–86. DOI: 10.1109/ICCIT52419.2022.9711620.

[53] Yinhe Liu et al. “Scale-aware deep reinforcement learning for high resolution remote sensing imagery classification”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 209 (2024), pp. 296–311. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2024.01.013. URL: https://www.sciencedirect.com/science/article/pii/S0924271624000224.

[54] Wen Nie et al. “A Deep Reinforcement Learning-Based Framework for PolSAR Imagery Classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–15. DOI: 10.1109/TGRS.2021.3093474.

[55] Matt L. Sampson and Peter M. Melchior. “Spotting Hallucinations in Inverse Problems with Data-Driven Priors”. In: *Machine Learning for Astrophysics*. July 2023, 29, p. 29.

[56] Sayantan Bhadra et al. "On Hallucinations in Tomographic Image Reconstruction". In: *IEEE transactions on medical imaging* PP (May 2021). DOI: 10.1109/TMI.2021.3077857.

[57] Chulkyun Ahn and Jong Hyo Kim. "AntiHalluciNet: A Potential Auditing Tool of the Behavior of Deep Learning Denoising Models in Low-Dose Computed Tomography". In: *Diagnostics* 14.1 (2024). ISSN: 2075-4418. DOI: 10.3390/diagnostics14010096. URL: https://www.mdpi.com/2075-4418/14/1/96.

[58] Zhenzhen Liu et al. "Unsupervised out-of-distribution detection with diffusion inpainting". In: *Proceedings of the 40th International Conference on Machine Learning*. ICML'23. JMLR.org, 2023.

[59] Mark S. Graham et al. "Denoising diffusion models for out-of-distribution detection". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023, pp. 2948–2957. DOI: 10.1109/CVPRW59228.2023.00296.

[60] Wenbo Li et al. "MAT: Mask-Aware Transformer for Large Hole Image Inpainting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[61] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381.

[62] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[63] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Version 0.3.0. Aug. 2020.

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

[65] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG].

[66] Richard Zhang et al. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*. 2018. arXiv: 1801.03924 [cs.CV].