NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

JOANA CARDEIRA INVERNO

BSc in Micro and Nanotechnology Engineering

BUILDING FOOTPRINT POLYGONS FOR THE REGION OF PORTUGAL

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

BUILDING FOOTPRINT POLYGONS FOR THE REGION OF PORTUGAL

JOANA CARDEIRA INVERNO

BSc in Micro and Nanotechnology Engineering

Adviser: Luís Oliveira
Associate Professor, NOVA University Lisbon

Co-adviser: Henrique Oliveira
Coordinator Professor, Instituto Politécnico de Beja

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

BUILDING FOOTPRINT POLYGONS FOR THE REGION OF PORTUGAL

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.5).

Acknowledgements

I want to express my deepest gratitude to all those who have contributed to the shaping of this thesis, which symbolizes the culmination of my academic journey.

I am thankful to my thesis advisors, Professor Luis Oliveira and Henrique Oliveira, for their invaluable guidance, mentorship, and expertise throughout this project. Having Professor Henrique Oliveira as my co-adviser has been an immense privilege, and I am particularly grateful for the trust and confidence he has placed in my work, allowing me to find my own path and supporting me whenever necessary.

I extend my appreciation to Direção Geral do Território (DGT) for providing essential aerial images, which were crucial for developing datasets and training deep learning models.

I also wish to acknowledge Nova SST for providing me with invaluable knowledge, fostering meaningful friendships, and offering enriching experiences that have enabled me to surpass my own limitations and beliefs. Throughout this academic journey, I have had the opportunity to interact with inspiring professors and supportive colleagues who have broadened my perspective and contributed to my personal and professional growth as an engineer.

Lastly, I would like to express my deepest gratitude to my family and friends for their unwavering support, encouragement, and love. Their belief in me has been a constant source of strength and motivation, and I am forever grateful for their presence in my life.

”

“The man who moves a mountain begins by carrying away small stones.”

— Confucius

Abstract

Accurate delineation of building footprints is crucial for developing comprehensive maps in rapidly expanding urban areas. These maps play a vital role in understanding population distribution, delivering public services, and supporting fields like urban planning and disaster management. However, the diverse and complex nature of urban environments presents significant challenges for automated building detection.

This study explores the integration of two Deep Learning architectures, ResUNet and SwinUNet, within a system designed to accurately identify and delineate building footprints from aerial imagery. A novel dataset of aerial images from a region in Portugal, that comprises the districts of Santarém, Leiria, Castelo Branco, and Portalegre was created as the primary data source, complemented by two pre-existing datasets for validation. The research investigates several factors affecting performance, such as the impact of different datasets and model architectures, the integration of RGB and false-color imagery, and the models' generalization capability across various datasets and regions.

Following post-processing, the ResUNet architecture, trained on pre-existing datasets, outperformed state-of-the-art methods, achieving Recall, Precision, IoU, and F1-Score values of 86.44%, 89.66%, 76.61%, and 88.02% on the Inria Dataset, and 94.20%, 95.68%, 90.36%, and 94.93% on the WHU Dataset, respectively. For the novel dataset, the model achieved values of 75.64%, 57.90%, 49.19%, and 65.59%, respectively, due to the presence of imperfect footprints within the dataset, which significantly impacted the model's results. However, a qualitative analysis revealed that the model successfully detected most buildings in the images, suggesting a discrepancy between the metrics and the visual results.

Overall, the results highlight the effectiveness of the proposed approach for building footprint extraction from aerial images and demonstrate the models' applicability to diverse datasets and geographic regions.

Keywords: Building Footprint Extraction, Deep Learning, Object Segmentation, Convolutional Neural Networks, Vision Transformers

Resumo

A delimitação exacta de áreas edificadas é essencial para a elaboração de mapas abrangentes de áreas urbanas em rápida expansão. Estes mapas são cruciais para compreender a distribuição da população, a prestação de serviços públicos e apoio a domínios como o planeamento urbano e gestão de catástrofes. No entanto, a natureza variada dos ambientes urbanos coloca desafios significativos à deteção automática de edifícios.

Este estudo investiga a utilização de duas arquitecturas de *Deep Learning*, ResUNet e SwinUNet, integradas num sistema concebido para identificar e delinear com precisão áreas edificadas através de imagens aéreas. Para tal, um novo *dataset* composto por imagens aéreas de uma região de Portugal foi criado como fonte primária de dados e complementada por dois conjuntos de dados adicionais pré-existentes para validação de resultados. Este trabalho explora vários factores que podem influenciar os resultados, incluindo o impacto de diferentes *datasets* e arquitecturas de modelos, a integração de imagens RGB e de falsa-cor, bem como as capacidades de generalização dos modelos em diversos conjuntos de dados e regiões.

Após o pós-processamento, a arquitectura ResUNet, treinada nos conjuntos de dados pré-existentes, superou os métodos do estado da arte, alcançando valores de *Recall*, *Precision*, *IoU* e *F1-Score* de 86.44%, 89.66%, 76.61% e 88.02% no conjunto de dados Inria, e 94.20%, 95.68%, 90.36% e 94.93% no conjunto de dados WHU, respectivamente. Para o novo conjunto de dados, o modelo obteve valores de 75.64%, 57.90%, 49.19% e 65.59%, respectivamente, devido à presença de áreas edificadas com contornos imperfeitos, que afectaram significativamente os resultados do modelo. No entanto, uma análise qualitativa revelou que o modelo detectou com sucesso a maioria dos edifícios nas imagens, sugerindo uma discrepância entre as métricas e os resultados visuais.

No geral, os resultados destacam a eficácia da abordagem proposta para a extração de áreas edificadas a partir de imagens aéreas e demonstram a aplicabilidade dos modelos a diversos conjuntos de dados e regiões geográficas.

Palavras-chave: Extração de Áreas Edificadas, *Deep Learning*, Segmentação de Objetos, *Convolutional Neural Networks*, *Vision Transformers*

Contents

List of Figures

List of Tables

Acronyms

ANN Artificial Neural Network *(pp. 4, 5)*

CNN Convolutional Neural Network *(pp. 6, 7, 9–12, 18, 19, 30, 38, 43, 51, 52)*

DGT Direção Geral do Território *(pp. 2, 22, 23, 25, 27, 29, 35, 38, 40, 42–44, 46–51)*

GPU Graphics Processing Unit *(pp. 33, 38)*

IoU Intersection over Union *(pp. 8, 46)*

MLP Multilayer Perceptron *(p. 12)*

NDVI Normalized Difference Vegetation Index *(p. 24)*
NDWI Normalized Difference Water Index *(p. 24)*
NIR Near-Infrared *(pp. 24–26, 40, 50)*
NIR-RG Near-Infrared Red Green *(pp. viii, 26, 29)*

PNG Portable Network Graphics *(pp. 25, 28)*

RGB Red Green Blue *(pp. viii, 26, 29, 38–40)*

TFW World File *(pp. 23, 25)*
TIFF Tagged Image File Format *(pp. 23, 27)*

VHR Very High-Resolution *(pp. 1–3)*

1

Introduction

1.1 Context and Motivation

Portugal, located on the westernmost tip of Europe, spans a total area of 92,152 km², characterized by diverse land use and population distribution patterns. As presented in Figure 1.1, over fifty percent of Portugal's territory comprises forests and agricultural settings, with the population predominantly concentrated in metropolitan regions such as Lisbon and Porto, as well as along the western and southern coastlines [1, 2].

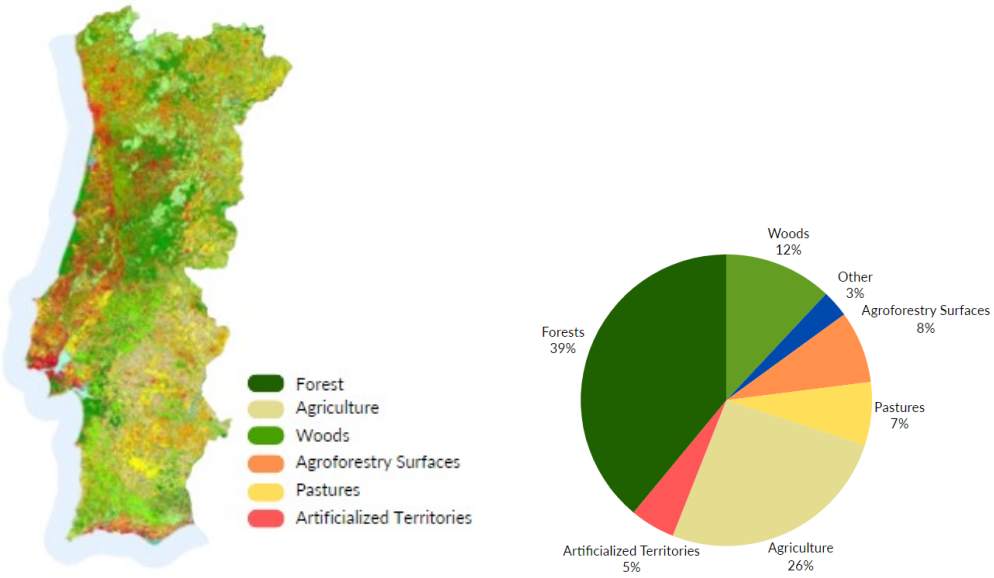
Accurate delineation of building footprints is vital for producing comprehensive maps, especially in rapidly growing urban areas. These maps play a crucial role in understanding population distribution, providing essential services, and supporting various fields such as urban planning and disaster management [3–5].

The significance of this task has led to the development of numerous methodologies for building detection over the past few decades, many of which rely on Remote Sensing technologies. These technologies enable researchers to understand, analyze, and remotely monitor various activities from considerable distances. Recent technological advancements have facilitated the acquisition of Very High-Resolution (VHR) aerial images, typically characterized by spatial resolutions ranging from sub-meter to a few meters. Such resolutions provide detailed ground information with precise delineation of urban structures, enabling more accurate and efficient building detection [5–7].

However, manually creating building footprints for large-scale maps is impractical due to its difficulty, time-consuming nature, error-proneness, and the requirement for expertise. Therefore, there is a growing need to employ automatic building detection methods [3, 8].

Despite the increasing interest in this field, existing strategies face significant limitations due to the diverse nature of urban environments. Factors such as spatial variability in building size, structure, complexity, background interference, varying illumination, occlusions, and shadows in images limit the efficiency of current methods in capturing building footprints [5, 6, 9].

When employing Very High-Resolution (VHR) aerial or satellite images, algorithms



(a) 2018 Land Use Chart (b) 2018 Land Use Statistics

Figure 1.1: Mainland Portugal Land Use Data for 2018 [2]

can be categorized into traditional handcrafted feature-based and Deep Learning-based methods. Traditional methods assess spectral, texture, and geometric information from images to distinguish buildings from the background. However, they heavily rely on human experts for feature extraction, which may overlook subtle or crucial features and may not be suitable for large-scale applications [10].

Conversely, Deep Learning methods represent a paradigm shift, achieving impressive performance improvements and often surpassing conventional methods on established benchmarks. These algorithms automatically learn detailed patterns, eliminating the need for explicit hand-crafted feature engineering and enabling consistent, reproducible results [11–13].

The Portuguese region studied in this work presents a diverse landscape, characterized by a mix of urban and rural areas. These characteristics make it an ideal location for studying building footprint extraction, as they provide a challenging environment for developing accurate models. The region's diversity allows for the evaluation of different methodologies and the exploration of their effectiveness in capturing building footprints across various settings.

1.2 Research Objectives

This study aims to develop a system capable of generating accurate building footprints for a specific region of Portugal, using VHR aerial images supplied by Direção Geral do Território (DGT). For this purpose, two distinct Deep Learning methods, ResUNet and

SwinUNet, will be implemented and evaluated to assess their effectiveness in building footprint extraction tasks across four datasets. This step will be followed by a polygonization process aimed at improving border precision.

1.3 Main Contributions

This study allowed for the creation of a robust system capable of accurately generating building footprints for a specific region of Portugal using VHR aerial images. The system also demonstrated the ability to generalize and produce accurate building footprints for other regions globally, as validated by the results obtained with the Inria and WHU datasets. It offers a thorough evaluation of two Deep Learning methods, trained across four different datasets. Additionally, the study investigates the influence of different datasets on the previously mentioned models on extraction accuracy, explores the impact of false-color images, and examines the importance of fine-tuning for model training. Furthermore, the study explores the effects of imperfect footprints on key metrics, providing valuable insights that contribute to advancing the field of building footprint extraction. Finally, the study proposes a polygonization process to improve border precision, to enhance the quality of extracted footprints.

1.4 Thesis Structure

This dissertation will be divided into five chapters: Introduction, Background and State-of-the-art Review, Proposed Methodology, Results and Discussion and Conclusions and Future Work.

The first chapter serves as the introduction to the dissertation. It will provide readers with an understanding of the context and motivation of the proposed problem, as well as a clear description of the objectives and an outline of the structure and organization of the document.

The second chapter is intended to present a comprehensive literature review. It serves both as the foundation for understanding the theoretical concepts used and exposition of the current State-Of-The-Art in the field.

The third chapter will detail everything about the proposed methodology used for creating the building footprints, from the theoretical basis of the components to the practical implementation of the system.

The fourth chapter will present a detailed analysis of the obtained results through the application of the proposed methodology. It will also compare these findings with other relevant methodologies and approaches.

The fifth and final chapter will offer a conclusive summary of the work developed during this dissertation. It will present the findings and outcomes of the project and propose potential directions for future improvements in this project.

2

Background and State-of-the-art

Image segmentation is a fundamental concept in computer vision that involves the labelling of individual pixels in an image, dividing it into meaningful regions or objects. It plays a pivotal role in various domains, like scene analysis, medical image processing or robotic perception. Over the years, a variety of segmentation algorithms have been developed. These methods have evolved from early approaches, such as thresholding, region growing, k-means clustering, and watershed techniques, to more advanced methods, including active contours, graph cuts, conditional and Markov random fields, and sparsity-based approaches. However, traditional feature-based methods have a shared limitation, they rely heavily on the quality of features extracted by human domain experts, which can often miss subtle or abstract features crucial for accurate image segmentation [11].

2.1 Deep Learning Methods for Image Segmentation

Deep Learning models have brought in a new perspective achieving impressive performance improvements, often outperforming conventional methods in well-recognized benchmarks. These new algorithms interact with problem-specific training data, to automatically learn intricate patterns, relationships and features from prior computations, extracting discriminative feature representations from massive databases, which allows them to handle large-scale, noisy and unstructured data. The automatic feature extraction also eliminates the need for explicit, handcrafted feature engineering, enabling these models to generate reliable and repeatable results, especially for tasks related to high-dimensional data. While Deep Learning models often require large, manually labeled datasets for effective training, which can be resource-intensive, this investment is justified by their ability to deliver superior performance when compared to other solutions [11, 12].

2.1.1 Artificial Neural Network

Deep Learning is a machine learning subset based on Artificial Neural Networks, ANNs, which mimic the structure and function of the human brain. They are composed of interconnected processing units called artificial neurons, which act as simplified mathematical

models. These neurons, also known as nodes, receive input signals and transmit them through weighted connections. The strength of these connections is adjusted during the learning process, allowing the network to learn from data.

The working principle of a neuron is presented in Figure 2.1. Incoming signals are multiplied by their corresponding weights and summed. This weighted sum is then passed through an activation function, which transforms the sum into the output signal [14, 15].

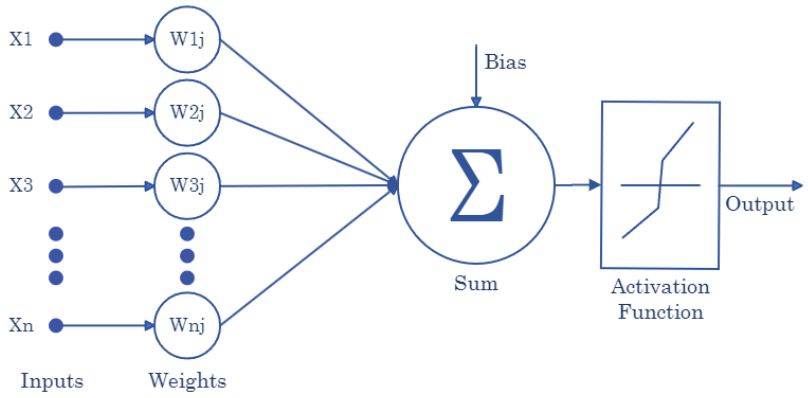


Figure 2.1: Schematic of the Working Principle of an Artificial Neuron

Activation Functions

The activation function plays a crucial role in the training performance of an ANN, since different activation functions result in different behaviours of the node, determining whether the neuron should be considered or not. This provides the necessary nonlinearity for the model to be able to learn complex representations. The choice of this function depends on the specific problem, architecture, and data distribution. Therefore, experimentation with different activation functions is common, since it can significantly impact the training speed, convergence and overall performance of the neural network. Some of the most common activation functions include the Sigmoid, Hyperbolic Tangent and ReLu (Rectified Linear Units) [16].

The Sigmoid function, is a non-linear function that maps the input values to an output from 0 to 1, causing output squashing and gradient vanishing, particularly in deep networks.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

The Hyperbolic Tangent function, similar to the Sigmoid function, squashes the input values in a range of –1 and 1. It has a steeper derivative and bounded output values, allowing for a wider range but tending to vanish due to its steeper gradient.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.2}$$

ReLU is a linear function with output values ranging from 0 to infinity, offering the ability to output true zero values, increasing representational sparsity in neural networks. It is the most common activation function used in deep neural networks as it helps prevent the vanishing gradient problem.

$$ReLu = max(0, x) \tag{2.3}$$

Deep Neural Networks

Deep neural networks are characterized by their multilayered architecture, Figure 2.2, where a series of interconnected neuron layers, with non-linear computations, enable the extraction and transformation of features. The determination of the number of layers and neurons is a manual process, along with setting parameters such as the learning rate and activation function, which collectively constitute the model's hyperparameters.

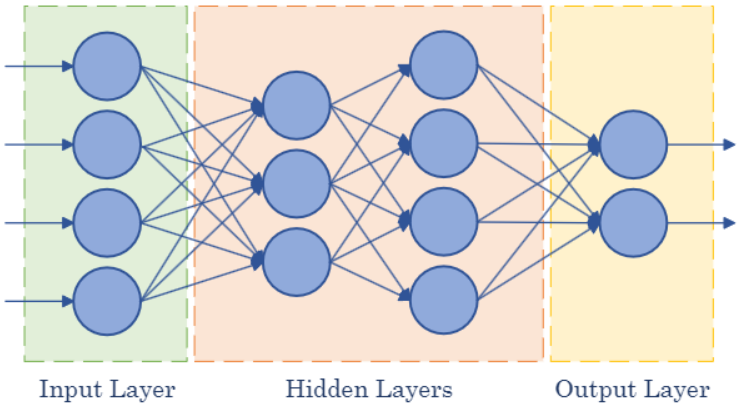


Figure 2.2: Schematic of the multilayered architecture of a Deep Neural Network

These layers are comprised of an input layer, which serves as the initial point where the model receives data input, and an output layer, responsible for producing the final result. Additionally, in between these layers, hidden layers are present, responsible for learning a non-linear mapping between input and output. These architectures present a hierarchical structure, where the layers are built upon the previous layers. Therefore, the layers closer to the data input are able to learn simple features, while layers closer to the output are able to learn more complex features, since they derive from the results obtained in the first layers [12, 17].

In the context of semantic segmentation Deep Learning models often employ an encoder-decoder architecture. The encoder compresses the input into a lower dimensional representation that captures the essential features and discards irrelevant information, while the decoder reconstructs this representation creating outputs that identify objects or regions of interest within the input image [18]. Architectures like Convolutional Neural Network (CNN)s and more recently, Transformers have been fundamental for the recent developments in this context.

Deep Learning Model Workflow

Before diving into the specifics of CNNs and Transformers, it is important to understand the general workflow behind the development of a Deep Learning model. This encompasses several sequential phases, each playing a crucial role in the model's predictive behaviour, Figure 2.3.

Figure 2.3: Sequential Workflow of a Deep Neural Network

The process starts with the acquisition of relevant data for the training of the model. Since these models automatically extract features from the dataset, the higher the amount of quality data, the better the model can be. This data is then pre-processed, where it is cleaned and formatted, the missing values are handled, and the features normalized. When necessary, the data is also augmented, by applying a series of strategies for the enlargment and enhancement of the dataset. This technique allows for the creation of new data from from an initial limited set, helping to minimize overfitting problems. During this phase there is also the creation of the Ground Truth masks. These masks serve as labelled references, indicating specific regions or objects that the model should learn to identify. After these masks are defined, and the data is properly seperated, the model architecture can be defined. This includes choosing the number and type of layers, the activation functions, and the overall structure of the model [19, 20].

After the architecture is defined, the model is trained. During the training phase, the parameters of the network undergo an iterative process. Initially, input data is fed through the network and the error between the predicted and actual output is computed. This difference is calculated through a loss function and serves as a guide to the optimization algorithm. The algorithm is employed to update weights in the opposite direction, effectively minimizing prediction errors. This iterative process continues until the minimum error is achieved, indicating the completion of the training phase [12, 21].

After the model is trained, its performance is tested through the test dataset. The images that were not used to train the model are essential to determine the generalization ability of the model and identify areas for improvement. If the results do not meet expectations, it is necessary to return to the training phase and adjust the parameters of the network [22].

There are several metrics that can be used to test the performance of the model, these should be chosen based on the specific objectives of the task to be performed. For semantic segmentation tasks, metrics such as Intersection over Union (IoU), F1-Score, Accuracy, and Precision are typically used, which compare the model's predictions to the reference

masks. Through the visualization and analysis of the results, it is possible to determine specific areas for improvement. Finally, after the model is trained and tested, it is ready to be deployed, and integrated into other applications [23].

Evaluation Metrics

The evaluation of a Deep Learning model's performance is a crucial step, allowing for the identification of areas for improvement and the determination of the model's generalization ability. Choosing appropriate metrics is essential and should align with the specific objectives of the task.

		Predicted class	
		Positive	Negative
True class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 2.1: Confusion Matrix

The common notation used to define these metrics is presented in Table 2.1, which serves as a valuable tool for summarizing a model's performance by categorizing predictions into four groups: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

True Positives represent the number of correctly predicted positive values, while True Negatives denote the number of correctly predicted negative values. Conversely, False Positives indicate the number of incorrectly predicted positive values, and False Negatives signify the number of incorrectly predicted negative values [24, 25].

For semantic segmentation, the most common metrics include recall, Precision, Intersection over Union (IoU), and F1-Score.

Recall measures the ratio of correctly predicted positive observations over the total number of actual positive observations, providing an indication of the model's ability to capture and identify positive samples. This metric is valuable when the cost of False Negatives is high.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Precision measures the ratio of correctly predicted positive observations over the total number of predicted positive observations, providing an indication of the accuracy of positive predictions. This metric is valuable when the costs of False Positives are high.

$$Precision = \frac{TP}{TP + FP} \tag{2.5}$$

IoU quantifies the overlap between predicted and ground truth bounding boxes. It is calculated by dividing the area of overlap by the area of union, providing a good indication of the model's performance.

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.6}$$

F1-score combines precision and recall, offering a single value representing a model's overall performance. It is particularly useful in situations with imbalanced class distributions.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.7}$$

2.1.2 Convolutional Neural Networks

Convolutional Neural Networks, or CNNs, are a specialized kind of neural network capable of processing data that is represented as a grid-like topology, such as an image. This architecture success has made them a fundamental technology in the field of image processing and computer vision, with a range of practical applications such as image classification, object detection and image segmentation [26, 27].

Compared to regular fully connected layers, CNNs exhibit some advantages. While fully connected layers establish connections between every neuron of the previous layer, CNNs take advantage of local connections reducing parameters and speeding up convergence. Moreover, these networks employ weight sharing, allowing a group of connections to share the same weights. Likewise, they incorporate a dimension reduction through the use of pooling layers. These layers leverage the principle of local image correlation to downsample an image, reducing the amount of data while retaining useful information [28].

CNN architectures vary, but in general, they are made up of convolutional and pooling layers that are organized into modules. These modules are typically stacked on top of each other to create a deep neural network. In addition to convolutional and pooling layers, these networks also incorporate fully connected layers, similar to those found in standard feedforward neural networks. These three types of layers, Convolutions, Pooling and Fully Connected Layers are the main building blocks of a CNN, where the first two perform feature extraction and the third maps the extracted features into a final output [13].

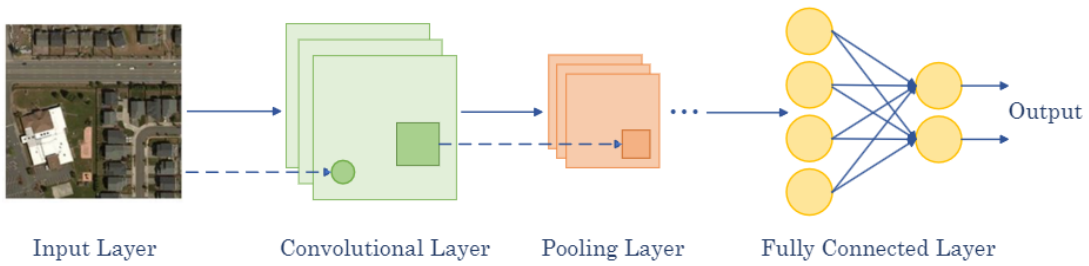


Figure 2.4: Schematic of a CNN pipeline

An example of a CNN architecture is illustrated in Figure 2.4. An input image is directly fed into the network, which then goes through a number of convolution and pooling layers. The representations obtained from these operations are subsequently directed into one or more fully connected layers, having the final output class dependent on the specific task the network is designed for [29].

Convolutional Layers

Convolution layers are a fundamental component of the CNN architecture. These layers employ a mathematical operation called convolution, a specialized kind of linear operation, which will capture and learn the feature representations of their input images. Convolution layers consist of learnable filters or kernels that act as parameters and output feature maps [26, 30].

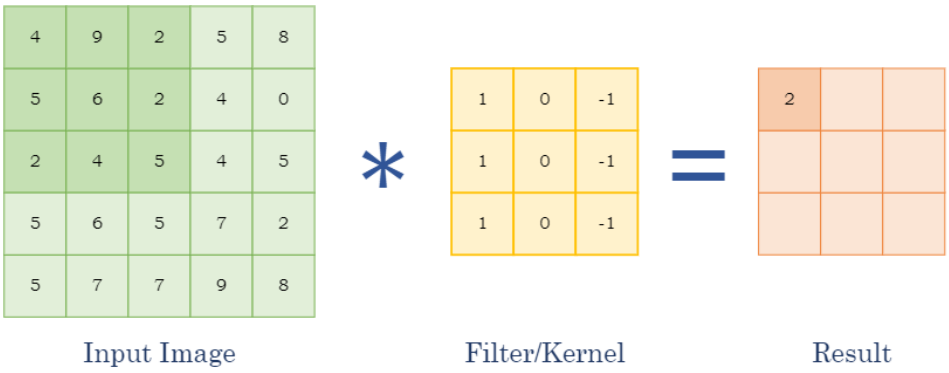


Figure 2.5: Schematic representing a Convolution Operation with a 3x3 Kernel

The mentioned operation involves sliding a small array of numbers, called a filter or kernel, across the input image, which is an array of numbers called a tensor. As this filter moves across the image, it computes the dot product between its values and the values present in the current receptive field, detecting patterns and features within the image. Each of the applied filters produces a feature map, which contains different aspects of the input data. A combination of these feature maps allows the network to learn low-level features such as edges and shapes and combine them to create more complex and abstract representations of the image [13]. Figure 2.5 presents a graphical representation of a convolution operation with a 3x3 kernel.

Some key hyperparameters that define a convolution operation are the size and number of filters. The stride, distance between two successive kernel positions, the padding and the activation function also define the convolution operation. A stride larger than one is sometimes used to achieve downsampling of the feature maps. An alternative is to perform a pooling operation [13].

Pooling Layers

A pooling operation has the intention of downsampling the feature maps, reducing their spatial resolution. This downsampling is crucial for achieving spatial invariance allowing the network to recognize features regardless of their precise location within the input. The primary example of this layer is to reduce computational complexity, while maintaining most of the dominant features. During pooling, a pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. For instance, the max pooling operation, Figure 2.6, reports the maximum output within a set neighborhood. Some other common pooling operations are average pooling, stochastic pooling and spatial pyramid pooling [30, 31].

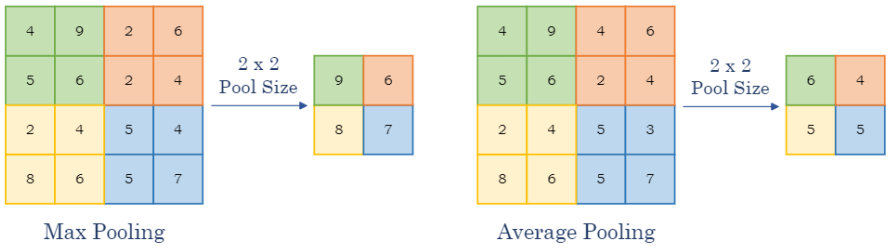


Figure 2.6: Schematic of Max and Average Pooling Operations

Fully Connected Layer

The final pooling or convolutional layer serves as input to the Fully Connected Layer, marking a transition to a structure similar to a conventional multiple-layer perceptron neural network. Here, all neurons from the previous layers are connected to the Fully Connected Layer and the hierarchical features extracted previously are made available for further analysis. The input to the Fully Connected Layer takes the form of a flattened feature map, a one-dimensional array of numbers. This flattening process serves as the foundation for learning and making predictions in various tasks [30, 31].

2.1.3 Vision Transformers

Transformer-based models, firstly introduced as an attention driven model for Natural Language Processing tasks, have shown promising results for various computer vision applications. These models replace the convolution operator with a self-attention mechanism to capture long-range dependencies. Therefore, instead of static filters being computed during the convolution operation in CNNs, self-attention dynamically computes its filters, allowing the model to focus on global context information. Moreover, these architectures present a great flexibility in handling vision tasks that require a large receptive field, high model capacity, less inductive bias, or grouping effect [32].

Like in CNNs, there are several variants of Vision Transformers, but they all follow the same underlying principle. They start by dividing an image into a grid of non-overlapping patches. The grid size is determined by dividing the image resolution by the patch size, a key configuration parameter for Vision Transformers. Smaller patches provide more detail but increase computational cost, making patch size selection a critical balance between precision and efficiency. Each patch is then embedded by applying a linear projection, creating a token. These tokens, which create a sequence of embedded patches, are then sent to the transformer encoder, where they will be processed by a series of multi-head attention and normalization layers. Finally, a Multilayer Perceptron (MLP) head receives the result from the encoder and outputs the input class. An example of a Transformer-based architecture is illustrated in Figure 2.7 [32, 33].

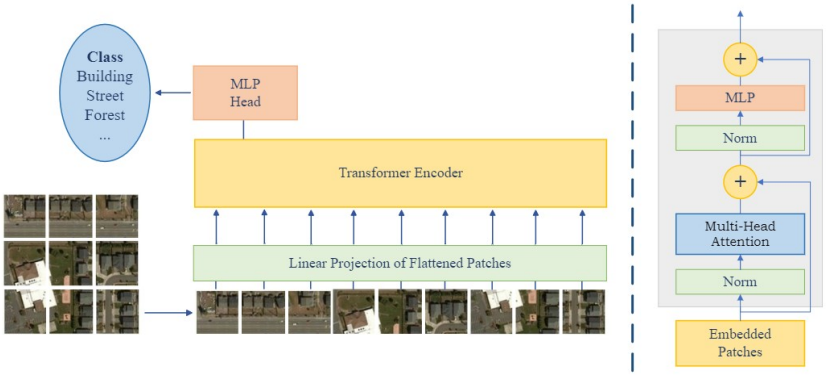


Figure 2.7: Schematic of a Vision Transformer Architecture and Encoder

Transformer Encoder

Each encoder module can be seen as a unit that processes the input information. This unit is comprised of various sub-modules, that apply attention mechanisms, normalization and feed-forward layers that capture long-range dependencies.

One of the key components within the transformer encoder is the Multi-Head Attention block. This block al lows for the connection of several Attention Heads, each equipped with its own matrix of adjustable weights. Other essential configurations for a Transformer include the number of attention heads, the depth (number of encoder layers), the embedding size, and the size of the feed-forward networks. These hyperparameters must be carefully chosen to balance model complexity and computational cost for a specific task. This characteristic allows the model to encode the interactions between all sequences of tokens [34].

Self-Attention

The key idea behind self-attention is to allow each token to learn self-alignment by aggregating global information from all other tokens in sequence [34].

This is achieved by introducing a compatibility function, with three learnable weight matrices that modify the input into queries (Q), keys (K) and values (V), and where the dimension of individual key/query vectors is represented by dk:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.8}$$

This attention mechanism outputs a dot-product that measures the representation similarity [34, 35].

To clarify, each token's representation determines whether a specific visual feature is present or absent at a particular location in an image, providing a visual interpretation or label for that position. This is achieved through a compatibility function, which measures the inter-token relations and determines how much a token should be influenced based on their visual content. Consequently, this function shifts the interpretation of individual tokens towards that of more compatible tokens in the input. As a result of these interpretations and adjustments, groups of tokens with similar visual characteristics start to form. Tokens that have similar representations, cluster together, facilitating the recognition of different elements in the image [35].

Since, the query, key and value have all the same source, originating from the previous block's output, the attention mechanism used in these blocks is referred to as "self-attention". When multiple blocks are concatenated simultaneously channel-wise to capture various complex interactions between different sequences of embeddings, this module is referred to as Multi-Head Attention. Each of the heads present in this module has their own learnable weight matrices allowing the model to capture a wide range of relationships and dependencies within the data [34].

Positional Encoding

Since Transformers do not have any convolutional or pooling layers, they lack the ability to capture spatial information. Therefore, although the self-attention mechanism allows the model to capture long-range dependencies, it does not take into account the relative position of the tokens. This absence of information about the relative position of tokens can be a significant limitation, especially in sequential data where the order is fundamental. To overcome this limitation, a Positional Encoding block is introduced, which, enriches the token embeddings with information about their relative position in the sequence [33].

This is achieved by adding a positional encoding vector to the input embeddings. This vector is calculated using sine and cosine functions of different frequencies, allowing the model to learn the relative position of each token in the sequence. For image data, this positional encoding embeds both spatial and sequential information, enabling the Transformer to better interpret relationships between image patches. The equations used to calculate the positional encoding are presented below, where pos is the position and i is the dimension [33].

$$PE_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.9}$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.10}$$

2.2 Post-Processing Techniques

Post-processing techniques are essential for refining the model's predictions and improving the quality of the output. In tasks such as semantic segmentation, these techniques help to mitigate challenges such as the loss of high-frequency details, noise, and artifacts in the output. The strategies described below encompass some of the most commonly used techniques in the field of semantic segmentation [36, 37].

2.2.1 Morphological Operations

Morphological operations are fundamental in filtering, feature enhancement, extraction, and compression tasks. This collection of non-linear operations modifies the geometric structure of an image, smoothing boundaries and filling gaps within segmented objects. Key morphological operations include dilation, erosion, opening, and closing, all of which depend on a structuring element that determines the geometry of the filtered object. Different shapes and sizes of structuring elements can be utilized to achieve various effects [38, 39].

Binary Dilation

Binary dilation adds pixels to the boundaries of objects, helping to connect disjoint segments and close small holes. This process can be visualized as sliding the center of the structuring element around the object's perimeter in the binary image. The dilated object is the original object expanded by the area covered by the structuring element [38, 39].

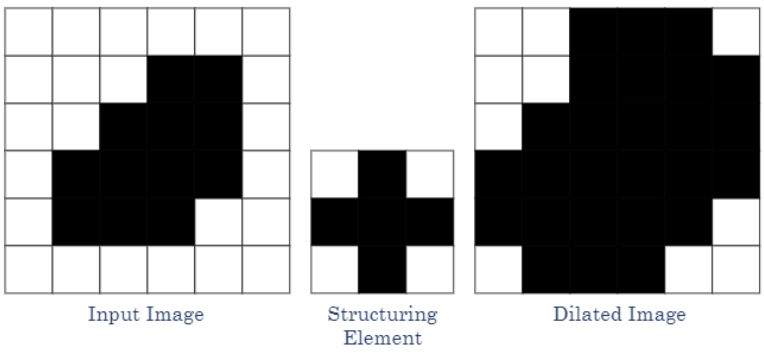


Figure 2.8: Binary Dilation Operation

Binary Erosion

Binary erosion removes pixels from the boundaries of objects, which helps separate touching objects and eliminate small noise. This process can be visualized as sliding the structuring element inside the object's perimeter in the binary image. The eroded object is the original object reduced by the area covered by the structuring element [38, 39].

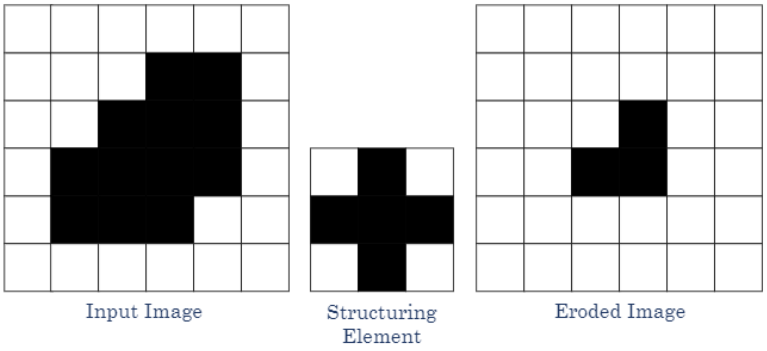


Figure 2.9: Binary Erosion Operation

Binary Opening and Closing

Two other fundamental morphological operators, opening and closing, are derived from dilation and erosion, further refining image features by smoothing and filling gaps, enhancing the overall quality and interpretability of the image.

The opening operation involves erosion followed by dilation. It smooths extrusions on binary contours, effectively removing small objects or noise. The closing operation consists of dilation followed by erosion. It smooths indentations in binary contours, filling small holes and gaps within objects [38, 39].

2.2.2 Image Smoothing Techniques

Image smoothing techniques are essential for refining and smoothing the boundaries of segmented objects, reducing jagged edges that can occur during the segmentation process. By choosing an appropriate smoothing method based on the specific characteristics of the noise and desired output quality, it is possible to reduce it and ensure an accurate delineation of the segmented images. Three of the most known methods are the Mean, the Median and the Gaussian filters, which work for both binary and gray scale images [40].

Mean Filter

The Mean filter is a simple and effective method for reducing noise in an image. This linear algorithm replaces the value of each pixel with the average value of the pixel and its neighbors. This process is repeated for every pixel in the image, effectively reducing

general noise and blurring sharp transitions at object boundaries. The Mean filter is particularly useful for reducing Gaussian noise, which can occur during the segmentation process [39, 40].

Median Filter

The Median filter is another effective method for reducing noise in an image. It replaces the value of each pixel with the median value of the pixel and its neighbors. This process is repeated for every pixel in the image, effectively reducing salt-and-pepper noise, which can occur during the segmentation process. The Median filter is particularly useful for preserving edges and fine details in the image, as it does not blur sharp transitions at object boundaries [39, 40].

Gaussian Filter

The Gaussian filter is a widely used method for reducing noise in an image. It replaces the value of each pixel with a weighted average of the pixel and its neighbors, with the weights determined by a Gaussian distribution. This process is repeated for every pixel in the image, resulting in a smoothed version of the original image. The Gaussian filter is particularly useful for reducing Gaussian noise, which can occur during the segmentation process [39, 40].

2.3 State-of-the-art Review

2.3.1 Datasets

The development of effective semantic segmentation models for the construction of these footprints requires large and diverse datasets, essential for training robust models that are capable of handling real-world scenarios. Recent advancements in remote sensing, have led to the creation of new datasets, each characterized by distinct classes and resolutions. The three main datasets that have emerged as prominent benchmarks for comparing results, are present in Table 2.2.

Name	Type	Bands	Resolution	Coverage	Image Size	Ref
Inria Aerial Image Labeling	Aerial	RGB	0.3m	810 km^2	1500x1500 px^2	[41]
Whu Aerial Building	Aerial	RGB	0.075m	450 km^2	600x600 px^2	[42]
Massachusetts Buildings	Aerial	RGB	1m	340 km^2	1500x1500 px^2	[43]

Table 2.2: Comparison of Three of the most used Datasets

The Inria Aerial Image Labeling Dataset covers varied urban settlements, ranging from densely populated areas to alpine towns. This aerial orthorectified colour imagery presents ground truth data for two semantic classes: building and not building [41].

The Whu Building Dataset, on the other hand, covers Christchurch, New Zealand and includes buildings with versatile architecture types, scales, and colours located in different urban and suburban areas. This aerial orthorectified colour imagery was manually edited and labelled into two classes: buildings and background objects [42].

Finally, the Massachusetts dataset covers the Boston area, covering most urban and suburban areas with buildings of all sizes, including individual houses and garages. Their aerial images were annotated by using reference maps and then hand-corrected to make the evaluations more accurate [43].

Based on the characteristics of these tree datasets, it becomes evident that there is great variability between them. The richness in diversity, both in terms of geographical coverage and characteristics of urban settlements of these datasets, despite their significance, introduces a layer of complexity when comparing results across them. The factors that contribute to the value of each dataset, also result in variations in the performance of semantic segmentation models when applied to different datasets. Divergencies in image resolution, size, coverage and annotation quality all contribute to the introduction of a substantial level of variability, which must be considered when doing cross-dataset comparisons.

To mitigate this challenge, it's important to be cautious when generalizing findings across datasets with different features and pre-processing steps. Therefore, similar datasets

should be considered when performing comparisons between different models [44].

2.3.2 Convolutional Neural Networks' Architecture

Over the years, CNNs have been the most widely used architecture for semantic segmentation tasks. These models have played a crucial role in the development of the field, introducing architectures such as AlexNet, GoogleNet, VGGNet, and ResNet. These architectures not only serve as benchmarks but also act as the foundation for innovative approaches that address common issues found in CNNs. These issues include limited receptive fields, problems associated with exploding and vanishing gradients, and the need for increased computational resources as network complexity grows [33, 45].

Since CNNs heavily rely on convolutions with strong inductive biases, they present a potential drawback: the loss of long-range dependencies. While the inductive bias contributes to the network's efficiency by promoting locality and translation invariance and allowing for effective extraction of local contextual information, it imposes a limitation — a restricted receptive field. This limitation makes CNNs less suited for capturing extensive global context information [46, 47].

Additionally, these networks may suffer from an exploding or vanishing gradient problem. During training, when backpropagation is used to adjust the model's parameters, the error is propagated backward to minimize the loss value. However, when the gradients diminish or approach zero rapidly, it impedes the model's ability to learn correlations across distant layers, presenting a vanishing gradient problem. Conversely, the exploding gradients problem arises when gradients grow uncontrollably, challenging the computation of meaningful updates to the model's parameters [48].

Numerous strategies have been developed to address these challenges, but none have provided a complete solution. Among these strategies, several architectures have emerged to enhance the capture of contextual information. Some models incorporate a residual squeeze and excitation pyramid pooling module, designed to fuse features across different levels and improve information exchange between channels. Others explore a criss-cross path, harvesting contextual information from all pixels along its trajectory, effectively aggregating dense contextual details with minimal computational and memory overhead. Additionally, some models utilize an atrous spatial pyramid pooling module, allowing for an effective incorporation of contextual details, while others employ multiscale dilation convolutions to extend their receptive fields and capture contextual information across different scales [49–53].

2.3.3 Vision Transformers' Architecture

Transformers, on the other hand, are a relatively new architecture in the field of computer vision. These models have been widely used in Natural Language Processing tasks, but their application to this type of tasks is still in its early stages. The first application of Transformers to a vision task was in 2018, when the Vision Transformer was introduced.

This model adopts a pure Transformer approach by directly applying a transformer to an input image divided into a sequence of patches [46, 47].

This architecture, however, presents significant limitations. The quadratic computational complexity, associated with the self-attention mechanism, makes the model computationally expensive, especially for large-scale datasets. Additionally, these models lack inductive bias, which is fundamental for the extraction of local contextual information, and are not translation invariant, as they process input sequences in a sequential manner without explicitly considering the spatial relationships between different regions [54].

In Vision Transformers, the self-attention mechanism operates globally, which comes at the expense of local information. The incorporation of inductive bias is applied sparingly throughout the model, making it harder for the model to build assumptions about the spatial relationships between different regions. This limitation is significant and makes the model heavily reliant on learning from diverse examples present in the training data. Therefore, this type of architectures require large datasets for training [54].

To overcome these limitations, several other models have emerged, each with its own unique characteristics. Some models adopt a hierarchical structure to achieve linear computational complexity, while others incorporate a spatial reduction layer and a progressive shrinking pyramid to reduce parameters. Additionally, certain models integrate local self-attention mechanisms, allowing the model to focus on specific regions within the input sequence [55–57].

2.3.4 Hybrid Models

Architectures like CNNs and Transformers have become undisputed leaders in the field of segmentation, delivering exceptional results and surpassing the state of the art in the creation of building footprints. As result of the limitations associated with the individual applications of these architectures, hybrid models, that leverage their respective strengths, have emerged holding great promise and potentially leading to the attainment of new state-of-the-art results [46, 47].

The adoption of hybrid models in building footprint detection remains limited, but the success of these structures in various domains is undeniable. These hybrid architectures, usually feature an encoder-decoder architecture with two parallel branches in the encoder. The transformer's branch efficiently captures global dependencies, while the CNN's handles the low-level spatial details. These branches are interconnected through a module designed to exchange key information and generate feature maps, which serve as the input for the decoder block, typically a CNN [58–63].

2.3.5 Final Remarks

The Tables 2.3 and 2.4 present a summary of the most relevant state of the art models for building footprints detection. The results presented in this table are based on the Whu Building Dataset and on the Inria Aerial Image Labeling Dataset.

Method	Architecture	IoU	Precision	Recall	F1	Ref
Deeplab v3+	CNN-based	89.61	94.68	94.36	94.52	[10]
Res-U-Net	CNN-based	89.44	94.34	94.51	94.42	[10]
Trans-UNet	Transformer-based	87.91	93.45	93.69	93.57	[64]
Swin-UNet	Transformer-based	89.50	94.53	94.38	94.46	[64]
BuildFormer	Hybrid	91.44	95.65	95.40	95.53	[65]
DSAT-Net	Hybrid	92.04	96.17	95.54	95.81	[64]

Table 2.3: Performances of different architectures on the Whu Dataset

Method	Architecture	IoU	Precision	Recall	F1	Ref
Deeplab v3+	CNN-based	76.80	87.37	86.40	86.88	[10]
Res-U-Net	CNN-based	76.91	87.69	86.22	86.95	[10]
Trans-UNet	Transformer-based	80.77	90.44	88.32	89.36	[64]
Swin-UNet	Transformer-based	76.27	86.87	86.21	86.54	[64]
BuildFormer	Hybrid	81.44	90.75	88.81	89.77	[65]
DSAT-Net	Hybrid	82.68	91.62	89.44	90.52	[64]

Table 2.4: Performances of different architectures on the Inria Dataset

The results present in these tables, will serve as a reference for the evaluation of the performance of the models created in this dissertation. As it can be observed, the hybrid models, DSAT-Net and BuildFormer, present the best results, outperforming the other models. This is a clear indication of the potential of hybrid models.

3 Methodology

To develop an automated system for building footprint extraction from high-resolution aerial images, the approach must be subdivided into several steps, starting with the preparation of the data and culminating in the creation of building footprints. Figure 3.1 presents a graphic representation of the methodology taken. Firstly, the original images are examined. This initial step enables the creation and refinement of the ground truth masks, which are essential for effectively training the developed model. Once the model generates predictions with satisfactory metrics, the data is post-processed to enhance the overall results. Upon completing this process for all datasets, conclusions can be drawn based on the findings.

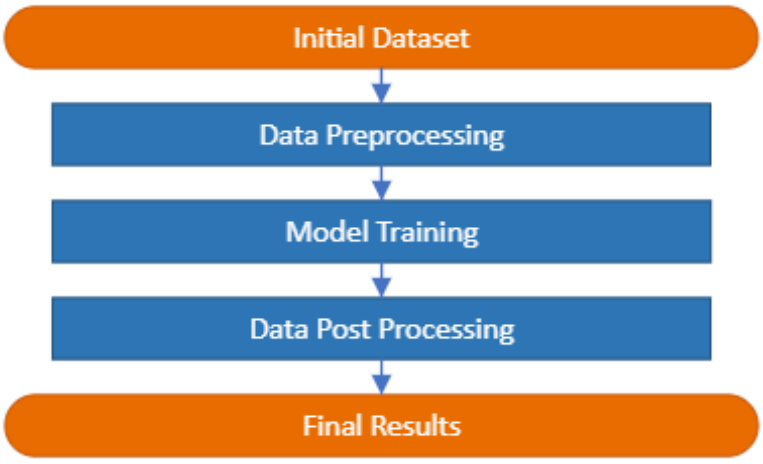


Figure 3.1: Methodology Overview.

3.1 Tools and Resources

This methodology is implemented using the Python programming language (version 3.11.5), leveraging widely used libraries and frameworks for Deep Learning and image processing. PyTorch (version 2.2.0+cu118) serves as the primary Deep Learning framework, and PyTorch Lightning (version 2.2.0) is used as a wrapper library to streamline

training and experimentation processes, abstracting away boilerplate code and promoting structured, scalable, and flexible development practices. Additionally, libraries such as OpenCV (version 4.9.0) will be employed for various image processing tasks, complementing the Deep Learning aspects of the methodology. The computational resources involved in this dissertation are presented in Table 3.1.

Component	Specification
Operating System	Windows 11
GPU	NVIDIA GEFORCE RTX 3060
CPU	12th Gen Intel Core i5-12400F, 2.5 GHz, 6-Core
RAM	16 GB 2400 MHz DDR

Table 3.1: Hardware Specifications

3.2 Data Preprocessing

As previously discussed, the primary purpose of this dissertation is to develop a system capable of creating building footprints within a specific region of Portugal. To achieve this goal, a dataset provided by DGT was utilized. This dataset comprises a comprehensive compilation of high-precision imagery across a region that encompasses the districts of Santarém, Leiria, Castelo Branco, and Portalegre.

Figure 3.2: Region of the DGT Dataset.

Two extensively used datasets, the Inria Aerial Image Labeling and the Whu Aerial Building datasets, were also used to facilitate the comparison and draw conclusions. These datasets serve as valuable benchmarks for evaluating several aspects of the study. However, prior to their use, it is imperative to preprocess and adapt the data so that it is compatible with the methodology. This section presents the detailed methodologies for preparing the datasets, outlining the steps taken to refine the data for effective integration into our study framework.

3.2.1 DGT Dataset

Exploratory Data Analysis

The dataset obtained from DGT comprises high-resolution images derived from the Carta Militar de Portugal series. This series divides the Portuguese map into manageable blocks at a scale of 1:25000. Each block is uniquely identified by a numeric code corresponding to its geographic location.

For each block, 16 images representing different sections are provided, offering detailed depictions of the terrain and infrastructure within the specified region of Portugal as of 2018. These images serve as the foundation for the dataset creation process. Accompanying each image are World File (TFW) files containing metadata for georeferencing purposes, including essential information such as the coordinate system, pixel size, and rotation parameters, ensuring accurate spatial alignment and referencing within the dataset.

In total, 58 tiles stored in the Tagged Image File Format (TIFF) and their corresponding TFW files were provided. Each TIFF file has a resolution of 0.25 meters per pixel and consists of four bands representing the Red, Green, Blue, and Near Infrared channels, with different brightness and contrast levels. With a default dimension of 16000 by 10000 pixels, each tile covers an area of 4x2.5 kilometers.

Tile Division and Sorting

They were divided into smaller, spatially preserved units to facilitate the manipulation of these tiles and mitigate computational demands. This approach was essential to handle the limited computational resources and ensure that each tile could be processed independently without exceeding memory constraints. Given the small sample size of images, two distinct methods were employed to augment the dataset's size and variability.

The first method involved dividing the images into a grid of 16x16 tiles, with each tile representing a specific section of the image and no overlap between adjacent tiles. The second method followed the same principle, but the starting point was shifted by half the width and height of the final image size, resulting in a 15x15 matrix of tiles. Each of these new tiles has dimensions of 1000x625 pixels, corresponding to an area of 250x156.25 meters. This overlap effectively increased the number of training examples, contributing to data augmentation.

After this step, the dataset contained 27,898 images. Given the predominantly green landscape in this area and the scattered distribution of urban settlements across the tiles, it became clear that further refinement was necessary to optimize the dataset for the task at hand. The tiles were, therefore, removed where there were no buildings. By eliminating these non-building areas, the dataset was refined to focus solely on relevant features, allowing the model to learn the characteristics of building footprints better. This optimization ensured that the model was trained on a dataset tailored explicitly to the task, ultimately improving its performance and the accuracy of the resulting building

footprints.

The quantity of images made manual sorting time-consuming. Therefore, an initial automatic sorting was performed to overcome this challenge. Analyzing the spectral indices using the bands of these images made it possible to filter a great number of images.

The Normalized Difference Vegetation Index (NDVI) considers the red and the Near-Infrared (NIR) bands and is used to assess vegetation health and biomass density. NDVI values range from -1 to +1, with higher values indicating healthier and denser vegetation, while lower values correspond to sparse or stressed vegetation or non-vegetated surfaces. The NDVI is calculated using the following formula 3.1 [66].

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}} \tag{3.1}$$

Conversely, the Normalized Difference Water Index (NDWI) considers the green and NIR bands and is used to highlight open water features. NDWI values range from –1 to +1, with higher values indicating bodies of water, while lower values correspond to land surfaces. The NDWI is calculated using the following formula 3.2 [66].

$$\text{NDWI} = \frac{\text{Green} - \text{NIR}}{\text{Green} + \text{NIR}} \tag{3.2}$$

Each image was processed, and the average NDVI and NDWI values were computed (Figure 3.3). These values were then used to classify the image as relevant or irrelevant based on predefined thresholds. The thresholds for relevance were defined based on empirical analysis, which indicated that suitable values for identifying areas with little vegetation or water presence were less than 0.2. If either of these metrics was less than 0.2, the image was classified as relevant. Finally, a visual inspection was performed to confirm and refine the dataset, filtering out the remaining non-relevant images. From the original 27,898 images, only 1,085 were deemed relevant for inclusion in the final dataset.

Figure 3.3: Example of the values of NVDI and NDWI for different types of images.

Creation of Ground Truth Masks

Ground truth masks are essential to train Deep Learning models for semantic segmentation. These masks serve as a reference, accurately depicting the location and shape of buildings present in the original images. The model learns to associate specific image features with building footprints by providing labeled data alongside the images. This is an exhausting and meticulous job that an expert must carry out to guarantee accurate and consistent results across the entire dataset.

However, the images provided by DGT were not accompanied by such masks, which presented a challenge in the training process. Therefore, the Microsoft *GlobalMLBuilding-Footprints* [67] dataset was utilized to address this issue. This extensive dataset containing over 999 million building footprint polygon geometries from various locations worldwide, provided in a line delimited GeoJSON format, was crucial for developing this work. The quality of the GeoJSON file varies across regions, including rural and urban areas, mountains, and plains. However, the performance for the European region is known and can be seen in Table 3.2.

Metric	Precision	Recall	IoU	Rotation Error
Value	94.3%	85.9%	65.1%	10.28 degrees

Table 3.2: Performance metrics for the *GlobalMLBuildingFootprints* Dataset.

The TFW files that accompanied the DGT images were used to extract the masks from this dataset. These files contained image coordinates for the top-left corner of each image, allowing for the deduction of the coordinates from the other corners. However, since the initial images were in the EPSG:3763 coordinate reference system, while the *GlobalMLBuildingFootprints* dataset required EPSG:4326, conversion of coordinates was necessary. The precision of this conversion depends on the accuracy of the transformation between the two coordinate systems, which could result in slight distortions, particularly when transforming pixel coordinates to geographical coordinates.

The coordinates of these rectangles were then input into a script, which generated the corresponding GeoJSON file containing all the building footprints within the specified regions. These extracted files exhibited different offsets due to differences in orthorectification methods. To overcome this, a manual inspection and adjustment of these coordinates was performed, which improved the alignment with the original images.

Finally, the original images are converted into Portable Network Graphics (PNG) format, while the masks are saved as binary images indicating the presence or absence of building footprints. Figure 3.4 illustrates the outcome of this process.

The False-color Dataset

The dataset provided by DGT, presented a fourth band the NIR band, which is uncommon in datasets used to detect building footprints. This characteristic presented an opportunity to explore the potential of this band in accurately detecting building footprints.

Figure 3.4: Example of the resulting images and corresponding masks for the DGT dataset.

In multispectral imaging, the bands are often correlated once they represent different aspects of the same underlying physical scene. By joining multispectral bands, some details unique to certain bands can be enhanced, potentially improving the overall quality and effectiveness of the data [68].

Therefore, to evaluate the potential benefits of the NIR band, a dataset was created with a spectral shift applied. In this modification, the Red band was replaced with the NIR band, the Green band was replaced with the Red band, and the Blue band was replaced with the Green band. Figure 3.5 depicts a comparison of the histograms of a RGB image and a NIR-RG image.

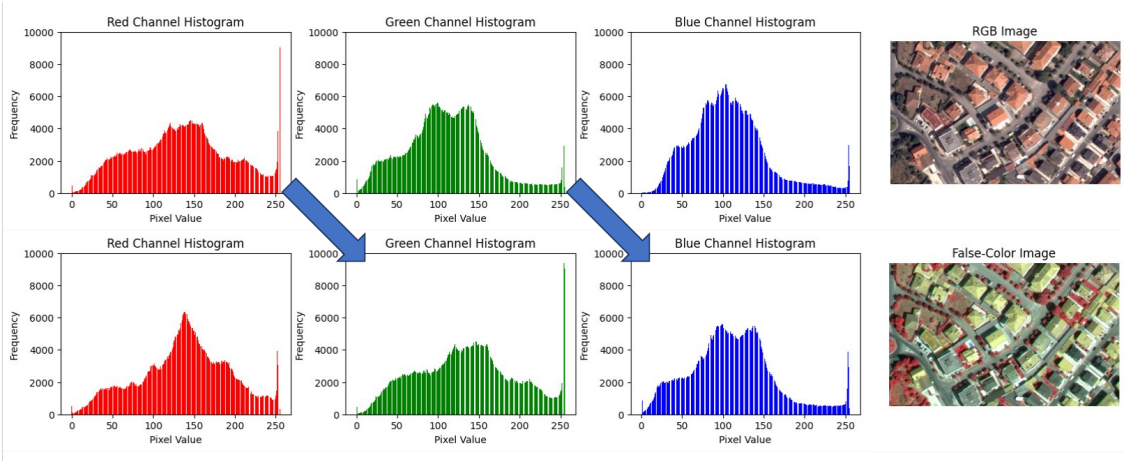


Figure 3.5: Comparison of Histograms: RGB vs. NIR-RG (NIR Replacement).

3.2.2 Inria Aerial Image Labeling Dataset

The Inria Aerial Image Labeling dataset [69] is a compilation of various urban settlements, ranging from densely populated areas to alpine towns. This open-source dataset has already been preprocessed and presented ground truth data for two semantic classes:

building and not building. However, before it can be used in the study, some characteristics must be tailored to match the DGT dataset.

These images consist of the three necessary bands: Red, Green, and Blue, and the resolution is similar to the DGT dataset, at 0.3 meters per pixel. However, these images were larger, measuring 5000 by 5000 pixels, and were in TIFF format. Thus, it was necessary to divide them into smaller images and convert them into PNG format. This division consisted of dividing the images into non-overlapping tiles, creating a 5x8 matrix with 40 images per tile, resulting in a total of 7200 images. An example illustrating the outcome of this process is presented in Figure 3.6.

Figure 3.6: Example of the resulting images and corresponding masks for the Inria dataset.

3.2.3 Whu Aerial Building Dataset

The Whu Aerial Building dataset [70] comprises images from Christchurch, New Zealand, and includes buildings with different architectural types located in different urban and suburban areas. This open-source dataset had already been pre-processed and presented ground truth data for two semantic classes: buildings and background objects. The dataset is structured into distinct partitions, as illustrated in Figure 3.7, the orange square represents the validation set, the blue square represents the training set, and the two red squares are the test set.

Despite the original dataset having images with a resolution of 0.075 meters, there was an option to choose a dataset of images that had already been up-sampled to have a ground resolution of 0.3 meters. Therefore, to tailor this dataset to match the characteristics of the DGT dataset, it was only necessary to increase the size and change the format of the images, as they originally had dimensions of only 512x512 pixels and were in TIFF format.

This increase was achieved by leveraging the way the images were partitioned. Since they were all from the same block, the next part of the image could be appended to the existing image. Thus, while the dataset remained the same size, the images were effectively

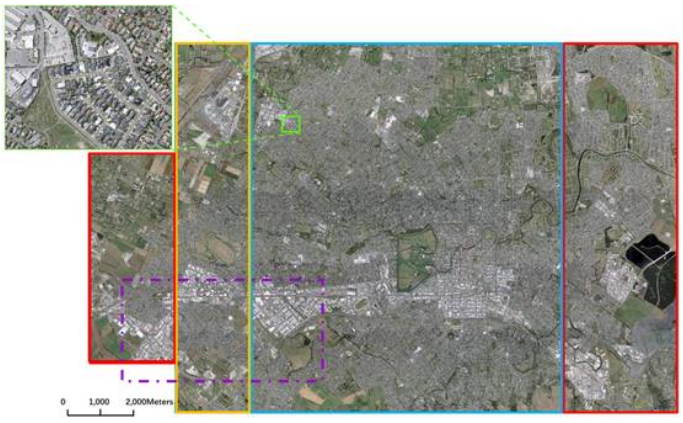


Figure 3.7: Overview of the covered area of the Whu Aerial Building dataset. Extracted from [70]

enlarged. The images were also converted to the PNG format. An example illustrating the outcome of this process is presented in Figure 3.8.

Figure 3.8: Example of the resulting images and corresponding masks for the Whu dataset.

3.2.4 Datasets Characteristics

After standardizing the images and ensuring they shared similar characteristics, all four datasets were organized using the same methodology. To facilitate the training phase, the images and corresponding masks were separated into three directories to create training, validation, and test datasets. These partitions comprised 70%, 10%, and 20% of the images and masks, respectively.

Images were stored in the PNG format, while masks were saved as binary images, with pixel values of 0 or 255 indicating the absence or presence of building footprints, respectively. Utilizing the PNG format ensures lossless compression, preserving image quality while significantly reducing file sizes, thereby facilitating efficient processing during semantic segmentation tasks. This reduced computational load enables faster model

training, evaluation, and inference, ultimately enhancing overall efficiency and saving time. Furthermore, this format is widely recognized and compatible with established standards, practices, and existing datasets [71].

An important consideration is the disparity in accuracy between the footprints for the Inria and Whu datasets compared to the polygons extracted from the *GlobalMLFootprints*. Additionally, it's worth noting that the DGT dataset contains significantly fewer images compared to the other two datasets. The influence of these characteristics will be examined in the upcoming section. The summary of the characteristics of each of the four datasets can be found in Table 3.3.

Datasets	**DGT (RGB)**	**DGT (NIR-RG)**	**Inria**	**Whu**
Bands	RGB	NIR-RG	RGB	
Image Number	1085		7200	8066
Spatial Resolution	0.25m		0.3m	
Image Type	Orthorectified Aerial Images			
Image Size	1000x625 pixels			
Training Ratios	70% Train, 10% Validation, 20% Test			

Table 3.3: Summary of the datasets characteristics.

3.3 Model Training

Several critical components must be considered to train a model effectively, each playing a crucial role in developing the best result possible. This process was divided into several blocks, allowing for a clearer understanding of the code and enabling selective modifications. Training was conducted within the Training file, while configurations and data management were handled in the Configuration file. The latter loads all the configurations required and imports the dataset from the Dataset Class and the model architecture from the Model Class. Separating these classes simplifies the replacement of the dataset or model architecture without requiring extensive code modifications.

Moreover, a dedicated evaluation file was created to assess the performance of the trained model. Here, the model, dataset, and Evaluator Class, which define various evaluation metrics, were loaded to conduct a comprehensive evaluation. Figure 3.9 illustrates the organitazional structure of the training.

3.3.1 Data Preparation

The Data Preparation stage ensures the dataset is appropriately processed and formatted for training. This critical step is implemented within the Dataset Class, which efficiently loads the images and their corresponding masks and applies necessary transformation methods to the dataset.

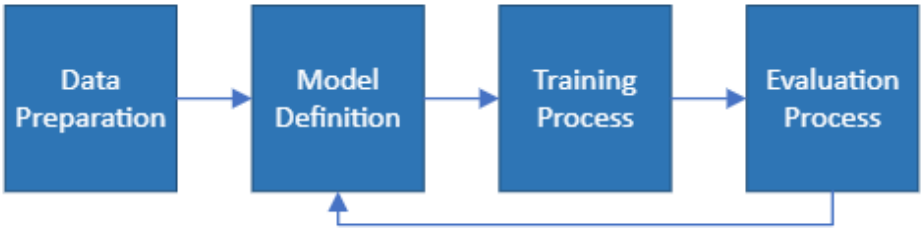


Figure 3.9: Block diagram of the training's organizational structure.

These transformations ensure uniform input dimensions across all samples, simplifying the training process and enabling efficient batch processing. Additionally, by normalizing the scale of the pixel values, the training process is stabilized, and the convergence accelerated.

Finally, the data augmentation horizontal and vertical flips were performed. These operations prove beneficial due to the frequently symmetrical nature of objects like buildings, roads, and vegetation, enhancing the model's generalization ability. Adjusting the brightness and contrast is another useful transformation in this context, since it helps the model adapt better to illumination changes. However, as the images already contained naturally variations in brightness and contrast, this method was not employed [72–74].

3.3.2 Model Definition

Determining the architecture used is essential before proceeding with model execution. For this work, two distinct models were selected: SwinUNet and ResUNet. The choice of these models was deliberate, aiming to assess and compare the performance of established CNN architectures, such as ResUNet, against emerging Transformer-based approaches, exemplified by SwinUNet, in the context of semantic segmentation tasks.

These architectures are inspired by U-Net [75], which is known for effectively handling segmentation tasks. The U-Net architecture is characterized by a symmetrical structure comprising a contracting path, where input data undergoes compression into a lower-dimensional representation, capturing essential features, and an expansive path that decodes this representation to generate outputs identifying objects or regions of interest.

The U-Net architecture, illustrated in Figure 3.10, employs repeated convolutional operations in the contracting path, followed by ReLU activation and pooling for down-sampling. Similarly, the expansive path mirrors this structure but employs up sampling instead. Skip connections are introduced to preserve detailed spatial information during up-sampling, linking corresponding layers between the contracting and expansive paths.

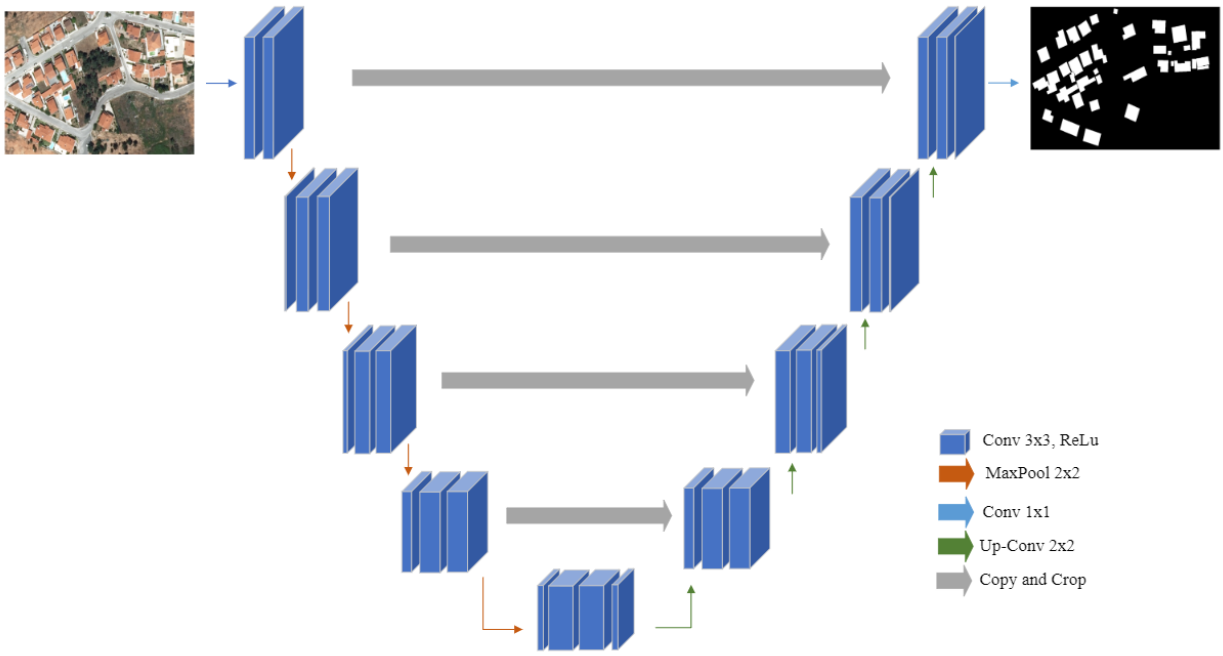


Figure 3.10: U-Net Architecture. Adapted from [75]

ResUNet

ResUNet [76] integrates residual connections within the U-Net architecture. These connections enable information propagation throughout the network, enhancing its ability to capture fine-grained details and long-range dependencies. Moreover, this characteristic helps mitigate the vanishing gradient problem and improve the model's overall performance by ensuring that relevant features extracted at earlier stages of the network are preserved and used in subsequent stages. The chosen backbone for the ResUNet architecture is ResNet-34, selected for its balance between performance and training speed.

This work used a Python library specializing in Neural Networks for Image Segmentation [77] to implement the ResUNet architecture. This comprehensive library offers encoders with pre-trained weights from ImageNet, allowing for better and faster model convergence. The ResUNet architecture is illustrated in Figure 3.11.

SwinUNet

SwinUNet replaces the convolutional basic unit of U-Net with a Swin Transformer block. This modification, as introduced by Hu Cao et al. [78], was one of the first pure Transformer-based U-shaped architectures and comprises a series of Swin Transformer blocks seamlessly integrated with layers for patch merging in the encoder and patch expanding in the decoder. This model leverages the shifted windows mechanism, enhancing the model's ability to capture spatial dependencies across different scales. Moreover, it allows

the model to focus on relevant spatial dependencies while minimizing computational complexity. The SwinUNet architecture is illustrated in Figure 3.12.

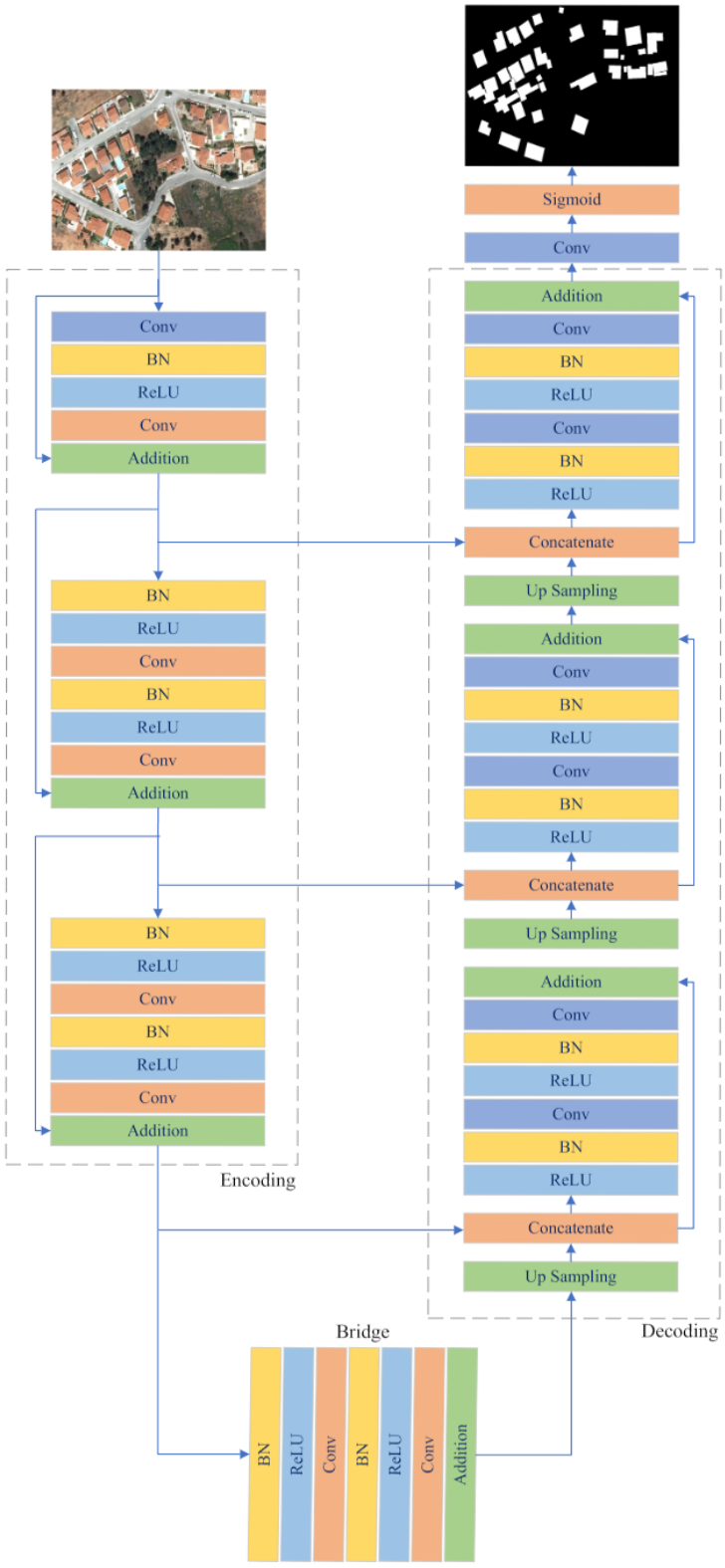


Figure 3.11: ResUNet Architecture. Adapted from [76]

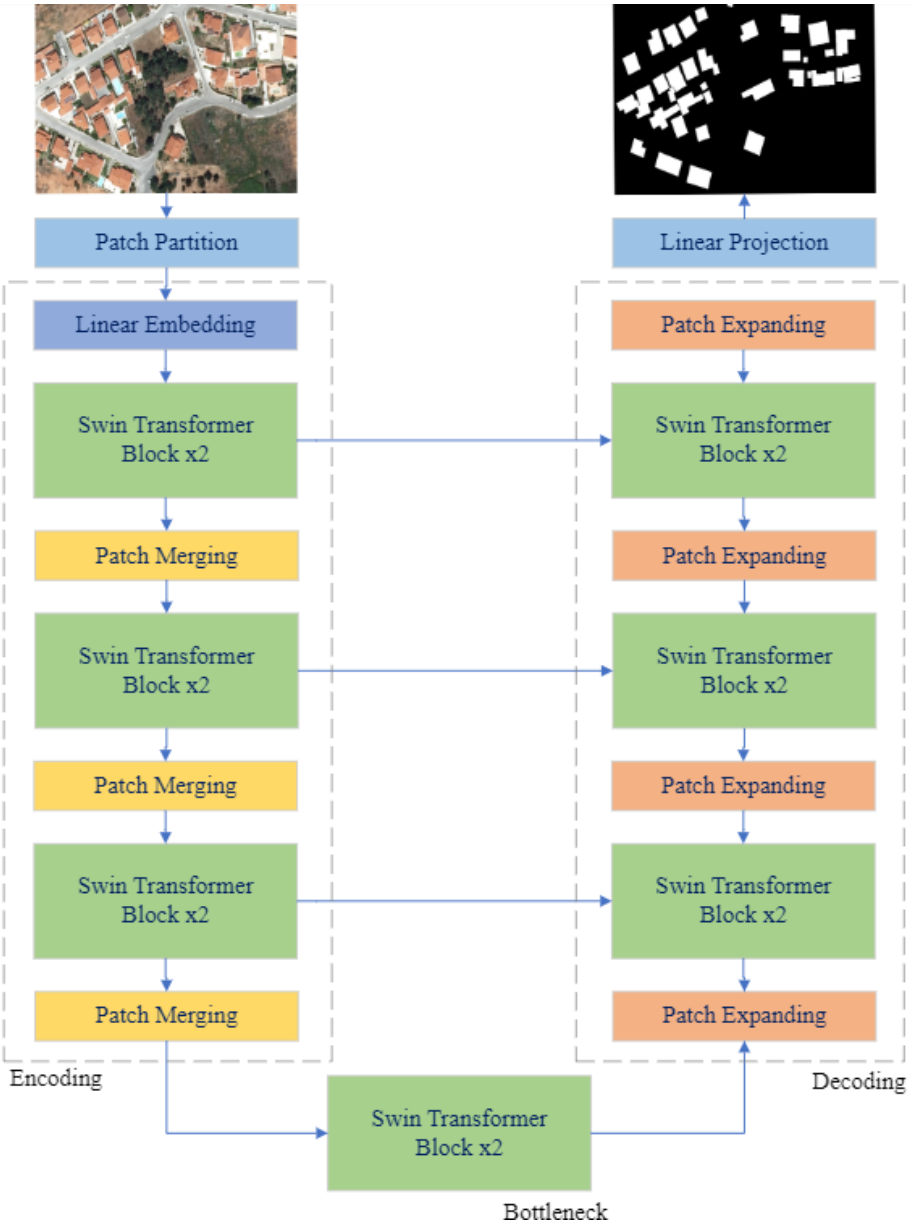


Figure 3.12: SwinUNet Architecture. Adapted from [78]

3.3.3 Training Process

Configuration File

Leveraging the PyTorch Lightning wrapper library [79], a more efficient and flexible development process was made possible. In the Configuration file, the chosen architecture, whether ResUNet or SwinUNet, is instantiated alongside the dataset loading process. Additionally, all the configurations necessary for development, such as the selected loss function, optimizer, and learning rate scheduler, are initialized here. These hyperparameters, along with other critical settings, such as the maximum number of epochs, logging frequency, and accelerator type (Graphics Processing Unit (GPU)), were carefully chosen

to optimize the training process. Table 3.4 provides a detailed overview of the initial values of these hyperparameters, offering insight into the training setup's configuration.

Hyperparameter	**Optimal Value**
Loss Function	BCEWithLogitsLoss
Optimizer	Adam
Learning Rate	0.0001
Batch Size	8
Max Number of Epochs	50
Learning Rate Scheduler	CosineAnnealingWarmRestarts
Patience	5

Table 3.4: Optimal Hyperparameters

During the training, PyTorch Lightning enables dynamic adjustments of hyperparameters and incorporates critical features like automatic learning rate scheduling and early stopping. The early stopping mechanism, defined by the patience variable, specifies the number of epochs with no improvement, after which training should be stopped, helping to prevent overfitting and enhancing model convergence.

Training Execution

The training execution is defined within the Training File. Here, a class that inherits the functionalities and features already provided by PyTorch Lightning allows for the customization of specific methods such as the forward pass, training step, validation step, and optimizer configuration.

In the main loop of the training execution, the Early Stopping and Model Checkpoint callbacks monitor the validation loss and save the best model checkpoint based on the validation performance. This enables the preservation of the best-performing model parameters while monitoring signs of overfitting or lack of progress. Seed values were also set to ensure result reproducibility.

A fine-tuning process was implemented to assess the influence of incorporating additional datasets on model performance. This process involves loading a pre-trained model from a specific checkpoint and continuing training with the new data or settings. Finetuning accelerates convergence and allows the model to leverage previously learned features, enhancing performance on the new task or dataset by initializing the model with the weights learned from previous training sessions.

Finally, a trainer object is instantiated, allowing for further customization of the training process. The model is then trained, and the best checkpoints and metrics, such as validation and training loss, are saved.

3.3.4 Evaluation Process

To be able to access the results and facilitate comparisons with other benchmark architectures and datasets, it is imperative to establish evaluation metrics that enable such comparisons. The Evaluator Class comprises all the functions necessary for a thorough evaluation, including metrics such as Recall, Precision, Intersection over Union, and F1-Score.

This class works with batches. For each batch, a confusion matrix is computed to compare the predicted and the ground truth mask, containing counts of true positives, false positives, true negatives, and false negatives. Each time an image from that batch is processed, the values from the confusion matrix update. The metrics are then calculated based on the confusion matrix values for each batch and saved in a list. At the end of the evaluation process, the average of these metrics is calculated, corresponding to the final performance of the model.

This process involves loading the previously trained model and assessing its performance on a separate test dataset. This dataset exclusively comprises images unseen by the model during training, thereby allowing for an evaluation of its generalization capabilities. All test images are loaded, and predictions are generated to evaluate the performance of this new dataset. These predictions represent probabilities indicating the likelihood of each pixel belonging to a particular class, ranging from 0 to 1, where 0 indicates low confidence, and 1 indicates high confidence.

Thus, a sigmoid function is used to create the corresponding binary mask, which allows for the conversion of the continuous probability values into binary values. Subsequently, the metrics are computed using the functions from the Evaluator Class and aggregated into a dictionary containing all test results. The final performance results are then derived by averaging across all images. The final results are visually inspected by plotting the images with the predicted mask overlaid.

Additionally, to determine whether the model's performance indicates overfitting, the test results are compared with previously calculated and saved validation results obtained during the training phase. This comparison provides insights into the model's ability to generalize to unseen data and ensures its robustness.

3.4 Post-processing

This work culminated in the creation of predicted building footprint polygon masks for the dataset provided by DGT. The process mirrors the evaluation step: an image is loaded, and a prediction is created. Subsequently, a sigmoid function is applied to create the binary mask, where each pixel is classified as belonging to a particular class based on its probability score.

The predicted masks are then post-processed to enhance the quality of the results. This post-processing step removes small artifacts and noise from the masks by applying a

mean filter to the binary mask. This filter replaces each pixel value with the mean value of its neighborhood, smoothing the mask and reducing the presence of minor isolated pixels.

Next, a dilation operation is applied, which expands the boundaries of the footprints and fills in the irregularities present in the masks. This operation involves sliding a structuring element over the binary mask and replacing each pixel value with the maximum value within the structuring element.

Finally, a conversion process was necessary since the initial predictions were not in polygon form. This involved identifying the contours of the predicted masks and approximating each contour to form polygons, using the OpenCV library. This library uses the Douglas-Peucker algorithm [80] to approximate two-dimensional curves, reducing the number of points while preserving the overall shape. The result was a new image with polygons representing the building footprints.

The performance of these post-processed polygons was assessed and compared with the previously predicted non-polygon masks. This comparison allowed for the evaluation of the effectiveness and impact of this post-processing step on the final predictions.

4

Results and Discussion

This work's primary objective was to create building footprints for a Portuguese region. However, several other conclusions were drawn and investigated throughout its development. Therefore, in addition to the detailed evaluation of several models' performance before and after post-processing, this section will discuss the influence and results of five other key aspects: (i) how different datasets and model architectures influence building footprint extraction; (ii) the models' ability to exploit complementary information from RGB and false-color images; (iii) the impact of using diverse datasets to train the models; (iv) the generalization abilities of the models; and (v) the impact of imperfect footprints on key metrics.

4.1 Performance of Different Model Architectures

The performance of the ResUNet and SwinUNet models was analyzed using different data distributions. The results of this evaluation are in Table 4.1, which will serve as a baseline for all the improvements made subsequently.

Model	Dataset	Recall	Precision	IoU	F1
ResUNet	DGT RealColor	67.15	78.40	56.67	72.34
	DGT FalseColor	67.34	77.66	56.42	72.13
	Inria	86.31	88.08	77.29	87.19
	Whu	94.07	95.47	90.05	94.76
SwinUNet	DGT RealColor	60.31	73.22	49.41	66.14
	DGT FalseColor	56.03	72.82	46.34	63.32
	Inria	58.00	78.18	49.94	66.57
	Whu	86.36	90.27	79.01	88.27

Table 4.1: Performance of ResUNet and SwinUNet on different datasets.

The performance metrics used to evaluate these models include Recall, Precision, IoU, and F1-Score. These metrics provide a comprehensive assessment of the model's ability to extract building footprints from high-resolution aerial images. They are essential for evaluating the models' performance and identifying areas for improvement.

The results of this evaluation indicate that the ResUNet model outperforms the SwinUNet across all datasets, achieving higher metrics. This performance disparity can be attributed to the architectural differences between the two models. While SwinUNet's attention mechanisms excel in capturing contextual information, ResUNet's architecture with residual connections shows more effectiveness in leveraging local features and preserving spatial information.

Moreover, it is clear that both models perform best on the Whu dataset, benefiting from its homogeneous nature and larger size, facilitating easier pattern recognition and effective learning of diverse patterns. Conversely, the Inria dataset presents more challenges due to its diverse landscapes and varying building types, leading to slightly lower performance metrics. The DGT datasets, which comprise images from a specific Portuguese region, present much lower metrics than the other datasets. The reasons for this behavior will be discussed in the following sections.

In addition to performance metrics, an analysis of the models' computational efficiency, encompassing factors such as training time and resource utilization, was conducted. The SwinUNet model required more epochs to converge compared to ResUNet, indicating potential differences in computational efficiency between the two architectures. In I, the training loss of both models is shown, with SwinUNet requiring more epochs to reach convergence.

Further analysis into resource allocation during training, particularly GPU usage, also revealed that SwinUNet exhibited higher GPU utilization despite processing smaller images, almost reaching 100% capacity. This issue suggests a higher computational demand during training, with potential scalability and resource allocation implications in practical deployment scenarios. In contrast, ResUNet maintained lower GPUutilization, around 80% of capacity, indicating a more efficient use of computational resources. These findings align with the theoretical understanding that transformer-based models, like SwinUNet, tend to be computationally more demanding than CNN-based models, such as ResUNet.

4.2 Impact of False-Color Images

An analysis of the models' performance on the false-color dataset was conducted to evaluate their ability to exploit complementary information from the NIR band in false-color images. The results of this evaluation are shown in 4.1.

When observing the performance metrics of both the ResUNet and the SwinUNet architectures, it is clear that they achieve superior performance when trained and tested on RGB images as opposed to false-color images. This is corroborated by conducting a

comparative analysis of the models' performance on the real-color and false-color datasets, as shown in Figures 4.1 and 4.2.

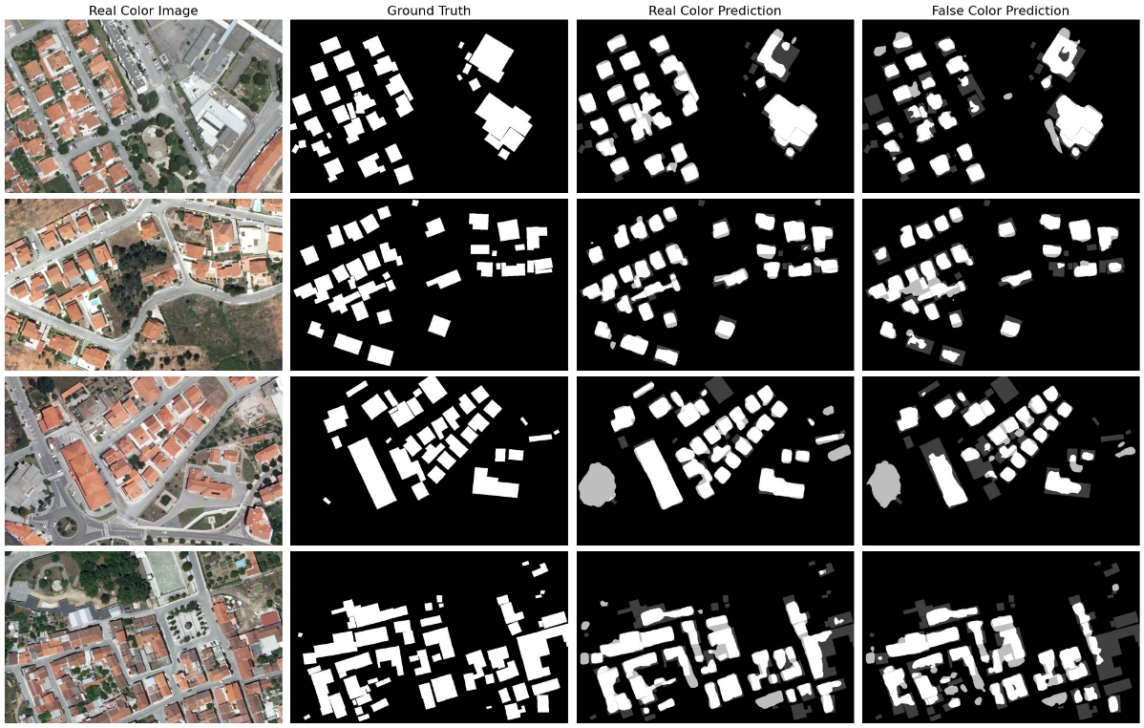


Figure 4.1: Performance of ResUNet model on the RGB dataset and False-Color dataset.

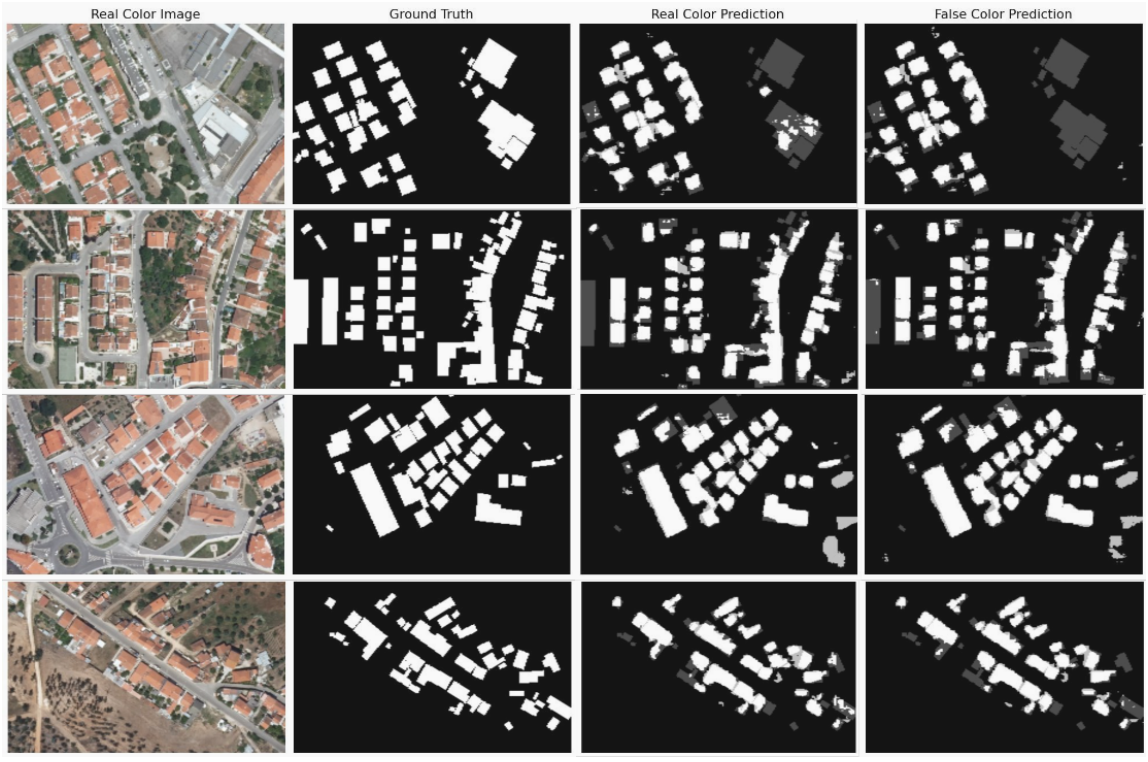


Figure 4.2: Performance of SwinUNet model on the RGB dataset and False-Color dataset.

These results suggest that the input data type influences the models' performance, with RGB images providing more relevant information for building footprint extraction tasks. The models' superior performance on RGB images can be attributed to several factors. RGB images contain more detailed and accurate information about the surface features, enabling the models to capture more precise representations of the underlying structures. In contrast, false-color images, which combine the NIR band with the RGB bands, provide less informative data and may hinder the models' ability to extract building footprints effectively. Moreover, the NIR band introduces additional information that may not be directly relevant to the task, increasing the complexity of the input data and leading to a decrease in generalization performance. Therefore, to achieve a better performance, it is advisable to use RGB images for training and testing, as they offer more comprehensive and informative data for the models to learn from. From here on, only the RGB images will be used to analyze of the models' performance.

4.3 Effect of Integrating Additional Datasets

Various combinations were explored to evaluate the impact of integrating additional datasets into the training pipeline on model performance. The quantitative evaluation of these architectures on the DGT real-color dataset is shown in table 4.2, the Inria Aerial Images dataset in table 4.3, and the Whu Building dataset in table 4.4. For a fair comparison, no post-processing methods were used. The false-color dataset was not studied since there was no significant improvement in performance when using it and because there was no data available that met the task requirements.

Model	Dataset	Recall	Precision	IoU	F1
ResUNet	DGT	67.15	78.40	56.67	72.34
	Inria + DGT	70.41	77.34	58.37	73.71
	Whu + DGT	68.95	**77.76**	57.58	73.07
	Inria + Whu + DGT	**71.90**	76.50	**58.89**	**74.12**
SwinUNet	DGT	60.31	73.22	49.41	66.14
	Inria + DGT	62.52	73.65	51.09	67.62
	Whu + DGT	63.32	73.31	51.46	67.95
	Inria + Whu + DGT	**63.48**	**74.61**	**52.20**	**68.59**

Table 4.2: Performance comparison of ResUNet and SwinUNet models for the DGT dataset with fine-tuning using Inria and Whu datasets.

Model	Dataset	Recall	Precision	IoU	F1
ResUNet	Inria	86.31	88.08	77.29	87.19
	DGT + Inria	86.04	88.13	77.11	87.07
	Whu + Inria	85.77	**89.10**	**77.63**	**87.40**
	DGT + Whu + Inria	**86.37**	88.32	77.52	87.33
SwinUNet	Inria	58.00	78.18	49.94	66.57
	DGT + Inria	61.55	75.77	51.45	67.90
	Whu + Inria	**73.72**	**82.26**	**63.61**	**77.75**
	DGT + Whu + Inria	72.28	81.19	61.92	76.47

Table 4.3: Performance comparison of ResUNet and SwinUNet models for the Inria Aerial Labelling dataset with fine-tuning using DGT and Whu datasets.

Model	Dataset	Recall	Precision	IoU	F1
ResUNet	Whu	94.07	95.47	90.05	94.76
	DGT + Whu	94.22	95.52	90.23	94.87
	Inria + Whu	94.20	**95.68**	**90.36**	**94.93**
	DGT + Inria + Whu	**94.26**	95.58	90.32	94.91
SwinUNet	Whu	86.36	90.27	79.01	88.27
	DGT + Whu	82.74	88.13	74.44	85.35
	Inria + Whu	**86.91**	**90.82**	**79.89**	**88.82**
	DGT + Inria + Whu	83.51	88.54	75.37	85.95

Table 4.4: Performance comparison of ResUNet and SwinUNet models for the Whu Aerial Building dataset with fine-tuning using Inria and DGT datasets.

The results show that integrating multiple datasets leads to improved model performance, as observed across all three datasets. The models trained on a combination of datasets consistently outperformed those trained on a single dataset, achieving higher recall, precision, IoU, and F1-Score values. This improvement in performance underscores the importance of exposing models to diverse data sources to enhance their ability to generalize and adapt to varying conditions. By training on a combination of datasets, the models gain a more comprehensive understanding of different building types, landscapes, and environmental conditions. This exposure facilitates learning robust representations, enabling the models to generalize effectively to unseen data and perform well in diverse real-world scenarios.

Additionally, the results demonstrate that the performance gains achieved by integrating multiple datasets are more pronounced when combining datasets with different characteristics. For instance, combining datasets with varying building types, landscape features, and annotation quality results in higher performance gains. However, ensuring that the datasets complement each other is crucial to maximizing the performance benefits.

4.4 Impact of Imperfect Footprints

The analysis of the impact of imperfect footprints on key metrics supports the previous conclusion. Across the Whu Buildings and Inria Labelling datasets, integrating either one consistently enhances the performance of the ResUNet and SwinUNet models. However, when incorporating the DGT dataset with another dataset, inconsistent performance improvements were observed.

This suggests potential mismatches or conflicts between the DGT and the other datasets that affect model performance. Differences in annotation quality, as reflected in Figure 4.3, are to blame and lead to suboptimal performance.

Figure 4.3: Annotations derived from the Inria Aerial Image Labeling dataset, the Whu Aerial Building dataset and the DGT Real Color Dataset.

Integrating datasets with imperfect footprints affects the models' performance However, there is a slight tolerance for imperfect data, as the models can still extract building footprints effectively. This adaptability is crucial for leveraging diverse datasets, even with varying levels of annotation quality. The models' ability to do this is particularly valuable in scenarios where obtaining high-quality annotations is challenging or resource-intensive.

Nonetheless, it is crucial to acknowledge that the quality of annotations still influences the models' performance. While the models can somewhat tolerate imperfect data, higher-quality annotations remain vital for achieving optimal performance. Thus, prioritizing efforts to enhance annotation quality and consistency is essential. The significant performance gap observed in the DGT dataset compared to others underscores the importance of improving annotation quality to ensure accurate and reliable results.

4.5 Optimized Performance Results

Table 4.5 presents the optimal performance results of ResUNet and SwinUNet models across different datasets. The performance is evaluated based on metrics such as Recall, Precision, IoU, and F1-Score. The results show that the models consistently perform best when trained with the CNN model, reiterating the advantages of using CNN-based architectures for tasks such as building footprint extraction from aerial images.

Model	Test Dataset (Train Dataset)	Recall	Precision	IoU	F1
ResUNet	DGT (Inria + Whu + DGT)	71.90	76.50	58.89	74.12
	Inria (Whu + Inria)	85.77	89.10	77.63	87.40
	Whu (Inria + Whu)	**94.20**	**95.68**	**90.36**	**94.93**
SwinUNet	DGT (Inria + Whu + DGT)	63.48	74.61	52.20	68.59
	Inria (Whu + Inria)	73.72	82.26	63.61	77.75
	Whu (Inria + Whu)	**86.91**	**90.82**	**79.89**	**88.82**

Table 4.5: Optimal performance results of ResUNet and SwinUNet models tested with different datasets.

The training dataset significantly influences the performance of both ResUNet and SwinUNet models, as evident from the varying performance metrics across different datasets. The results demonstrate the importance of selecting the appropriate training data to achieve optimal model performance.

The DGT dataset is the most challenging, with the models achieving the lowest performance metrics, with recall values of 71.90%, precision values of 76.50%, IoU values of 58.89%, and F1-Score values of 74.12% for the ResUNet model. This is due to the dataset's imperfect footprints and size, which hinder the models' ability to extract building footprints effectively. This dataset exhibits lower IoU values, suggesting difficulties in accurately delineating building boundaries and capturing spatial extents due to these imperfect footprints. Additionally, the smaller number of images and fewer buildings in the DGT dataset may limit the models' ability to learn diverse patterns and generalize effectively to unseen data.

The predictions for the DGT dataset are shown in Figure 4.4, where the models were able to extract building footprints. However, the predictions struggled to accurately delineate the building boundaries and capture the spatial extent of the structures, as confirmed by the lower IoU values. The poor building definition on the margins of the ground truth masks, where some buildings are lacking and some noise is presented when compared with the other datasets, have contributed to the lower performance metrics achieved.

Figure 4.4: Qualitative comparison of the models' predictions with the ground truth annotations for the DGT dataset. White: Overlap of Ground Truth and Prediction; Light Grey: Ground Truth; Dark Grey: Prediction

Performance on the Inria Labelling dataset lies between the DGT and the Whu Buildings datasets, indicating better quality annotations than DGT. The models could extract building footprints with higher performance metrics, such as recall values of 85.77%, precision values of 89.10%, IoU values of 77.63%, and F1-Score values of 87.40%, as shown in table 4.5. These results are consistent with the state-of-the-art results, which confirms the correct implementation of the two models.

By observing the models' predictions in Figure 4.5, it is possible to conclude that the models could accurately delineate the building boundaries and capture the spatial extent of the structures, as deduced from the performance metrics achieved, including from the higher value of IoU. Slight imperfections are still visible. However, the models were able to capture the majority of the buildings present in the images.

The last dataset, the Whu Buildings dataset, consistently outperformed the other datasets, surpassing the state-of-the-art results for that dataset on the ResUNet architecture. This architecture achieved recall values of 94.20%, precision values of 95.68%, IoU values of 90.36%, and F1-Score values of 94.93%, demonstrating the models' ability to extract building footprints with high performance metrics. This performance can be observed in the models' predictions in Figure 4.6, where the models could accurately delineate the building boundaries and capture the spatial extent of the structures, as deduced from the performance metrics achieved.

As previously mentioned, the training data influences the models' performance. The models achieve higher performance metrics in datasets with better-quality annotations and low variability in building shapes and sizes. The Whu Buildings dataset, characterized by high-quality annotations and consistent buildings, yielded superior results to the Inria Labelling dataset, which exhibited higher variation in landscape and building type. The DGT dataset presented the lowest performance metrics with imperfect footprints and

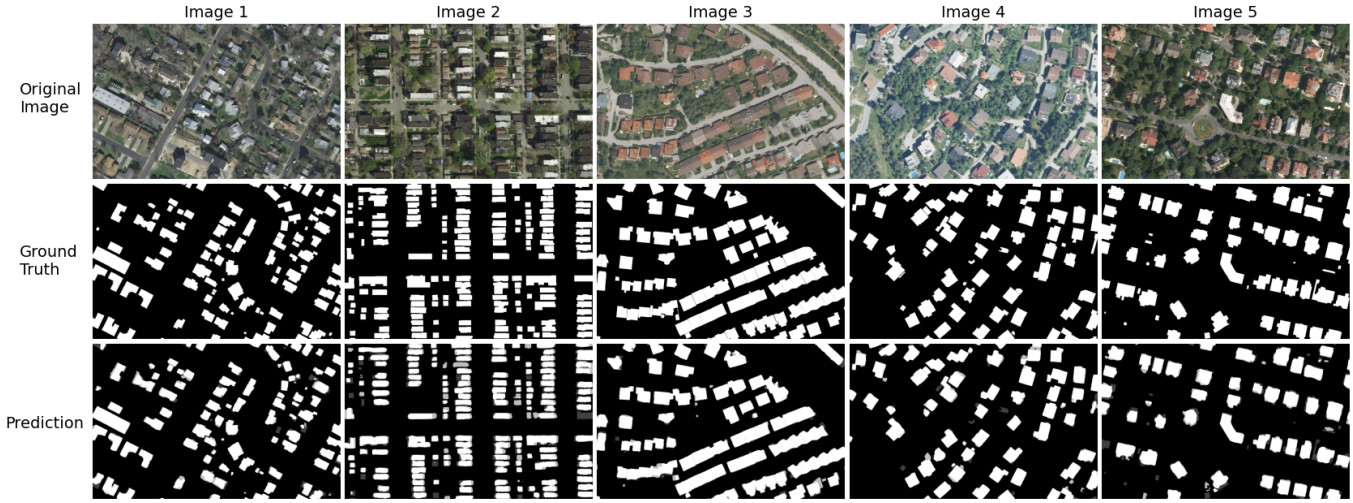


Figure 4.5: Qualitative comparison of the models' predictions with the ground truth annotations for the Inria Aerial Image Labeling dataset.

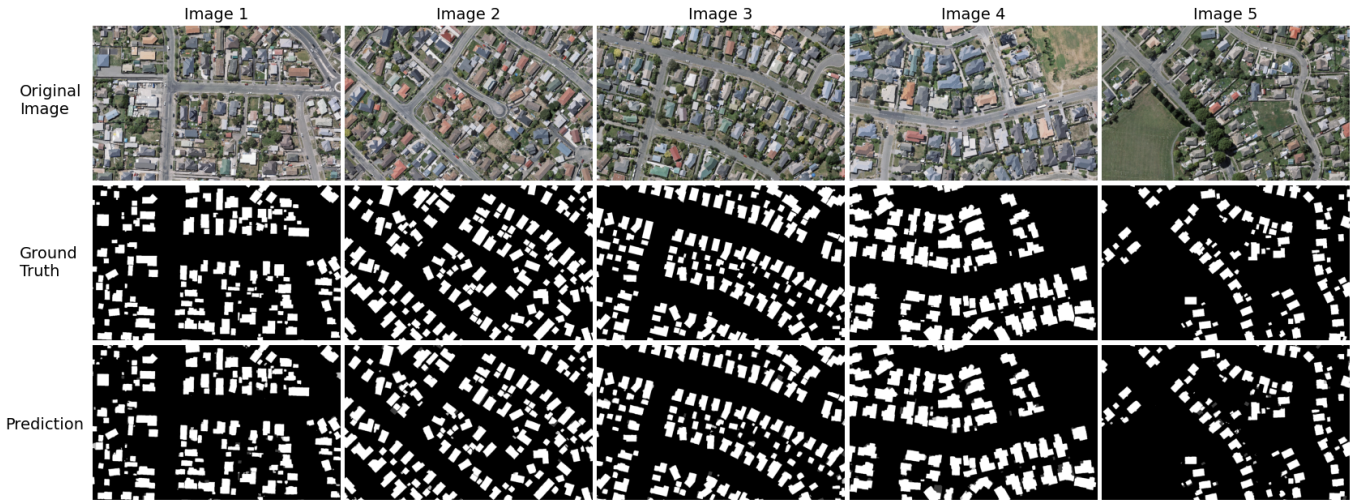


Figure 4.6: Qualitative comparison of the models' predictions with the ground truth annotations for the Whu Aerial Building dataset.

fewer images, highlighting the challenges associated with training models on datasets with inconsistent annotations and limited data.

4.6 Generalization Abilities of the Models

Another important aspect to evaluate is the generalization abilities of the models. To this end, the models with higher performance were tested on different datasets. The results of this evaluation are shown in Table 4.6.

The impact of the training dataset on the models' generalization abilities is evident from the varying performance metrics across different test datasets. The results demonstrate that the training data influences the models' performance, with the models usually achieving

	Train	DGT Dataset			Inria Dataset			Whu Dataset		
	Test	DGT	Inria	Whu	DGT	Inria	Whu	DGT	Inria	Whu
ResUNet	Recall	**71.90**	31.05	52.51	74.73	85.77	**88.38**	63.72	69.71	**94.20**
	Precision	76.50	90.63	**90.78**	57.76	89.10	**90.06**	60.79	84.46	**95.68**
	IoU	**58.89**	30.08	49.84	48.29	77.63	**80.53**	44.95	61.78	**90.36**
	F1	**74.12**	46.22	66.49	65.13	87.40	**89.21**	61.99	76.36	**94.93**
SwinUNet	Recall	**63.48**	15.67	59.57	61.33	73.72	**81.10**	32.88	27.51	**86.91**
	Precision	74.61	87.29	**86.68**	46.40	**82.26**	78.51	59.79	80.09	**90.82**
	IoU	52.20	15.32	**54.56**	35.90	63.61	**66.37**	26.16	25.74	**79.89**
	F1	68.59	26.55	**70.59**	52.74	77.75	**79.78**	40.66	40.92	**88.82**

Table 4.6: Generalization performance of ResUNet and SwinUNet models across different datasets.

higher performance metrics when tested on the same dataset they were trained on. This indicates that the models are more effective at extracting building footprints from data similar to the training data, as they have learned to recognize the patterns and structures present in that specific dataset.

The presence of imperfect footprints within the DGT dataset severely affects the models' generalization performance. Models trained on the DGT dataset exhibit lower performance metrics when tested on the Inria and Whu datasets. This issue highlights the difficulty generalizing these alternative datasets, particularly evidenced by lower IoU values. This low value of IoU indicates that the models struggle to accurately delineate building boundaries and capture the spatial extent of the structures when tested on the Inria and Whu datasets, which is a direct consequence of the imperfect footprints present in that dataset.

Conversely, the models trained on the Inria and Whu datasets, despite achieving slightly lower performance metrics when tested on the other datasets, indicate that the models could generalize effectively to unseen data and perform well in diverse real-world scenarios. This ability to generalize across different datasets demonstrates their robustness and adaptability to various terrains and urban environments.

A qualitative assessment of the models' predictions on various datasets contradicts the conclusions drawn from the performance metrics (Figure 4.7). The models could extract building footprints from the Inria and Whu datasets with optimal results when tested in the DGT dataset. The low performance metrics achieved are due to the imperfect footprints present in the DGT dataset. The inaccurate ground truth annotations within the DGT dataset hinder the models' ability to make accurate evaluations, contributing to the lower-performance metrics observed.

Clearly, the models trained with the Inria and Whu datasets, which are more homogeneous and contain higher-quality annotations, behave better than those trained with the DGT dataset. Despite presenting lower performance metrics, the models' predictions

Figure 4.7: Qualitative performance of the generalization performance of ResUNet models trained on the DGT, Inria and Whu datasets and tested on the DGT dataset.

are accurate and closely aligned with the ground truth annotations. This discrepancy highlights the importance of considering the quality of the annotations when evaluating the models' performance. The imperfect footprints within the DGT dataset may lead to lower performance metrics, but the models' predictions are accurate and reliable, as demonstrated by the qualitative results.

These datasets, as opposed to the DGT dataset, can capture the spatial extent of the structures and accurately delineate the building boundaries. Additionally, the borders of the buildings are well-defined and make a clear distinction between the buildings and the background, which is not the case for the models trained with the DGT dataset. Therefore, despite the low-performance metrics, the models' predictions are accurate and closely aligned with the actual building footprints, as shown in Figure 4.7.

4.7 Results After Post-Processing

The post-processing step can potentially refine the models' predictions, enhancing the quality of the extracted building footprints. As discussed in the previous section, despite presenting lower performance metrics, the models' predictions trained with the Inria and Whu datasets closely aligned with the actual building footprints. Therefore, for this work,

the top-performing model, which was trained initially for the Whu dataset, was selected for the post-processing step. The results can be seen in Figure 4.8.

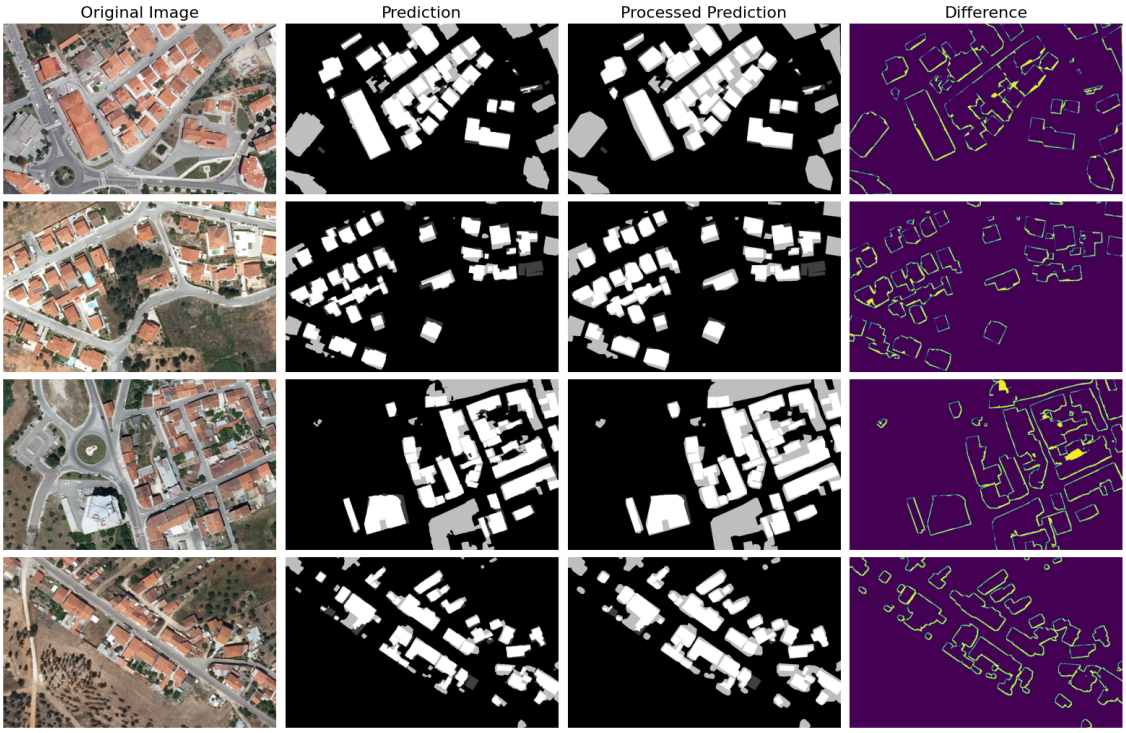


Figure 4.8: Results of the post-processing step applied to the top-performing model.

These results demonstrate the importance of incorporating post-processing techniques into the workflow to refine the model's predictions, correct imperfections, and enhance the quality of the extracted footprints. The post-processing step effectively removes noise, fills in gaps, and smooths out the building footprints, creating footprints that are more aligned with the desired output.

As mentioned, the performance metrics obtained for the DGT dataset were influenced by its imperfect footprints, which hindered the possibility of correctly evaluating the models' predictions. To confirm the efficiency of the post-processing step, the performance metrics of the models with perfect footprints, the Inria and Whu datasets, were analyzed. The results of this evaluation are shown in table 4.7.

	DGT Dataset		**Inria Dataset**		**Whu Dataset**	
	Before	**After**	**Before**	**After**	**Before**	**After**
Recall	74.73	**75.64**	85.77	**86.44**	94.20	94.20
Precision	57.76	**57.90**	89.10	**89.66**	95.68	95.68
IoU	48.29	**49.19**	77.63	**78.61**	90.35	**90.36**
F1	65.13	**65.59**	87.40	**88.02**	94.93	94.93

Table 4.7: Performance metrics before and after applying the post-processing method to the DGT, Inria, and Whu datasets.

The results show that the post-processing step effectively enhances the quality of the models' predictions, improving the performance metrics across the datasets. The models' predictions achieve higher performance metrics, confirming the efficiency of the post-processing step in enhancing the quality of the extracted building footprints.

4.8 Discussion

The results of this work demonstrate the importance of selecting appropriate training data to achieve optimal model performance. The performance of the models is significantly influenced by the quality of the training dataset, with higher performance metrics achieved when trained on datasets with high-quality annotations and low variability in building shapes and sizes. The presence of imperfect footprints within the DGT dataset severely affects the models' performance, leading to lower metrics and hindering the effective extraction of building footprints.

The models' generalization abilities are also influenced by the training data. Higher performance metrics are observed when models are tested on the same dataset they were trained on. Models trained on the Inria and Whu datasets, despite achieving slightly lower performance metrics when tested on other datasets, demonstrate their ability to generalize effectively to unseen data and perform well in diverse real-world scenarios. Additionally, the results underscore the importance of incorporating post-processing techniques into the workflow to refine predictions, correct imperfections, and enhance the quality of the extracted footprints.

Overall, when comparing the performance of the two benchmark models used in this work to state-of-the-art results (Table 4.8), it is evident that the models achieved competitive performance metrics, with the ResUNet model surpassing the state-of-the-art results for both datasets. This demonstrates the effectiveness of the models in extracting building footprints from aerial images and highlights their potential for real-world applications.

	Inria Dataset		**Whu Dataset**	
	State-of-the-Art	**Present work**	**State-of-the-Art**	**Present work**
Recall	86.22	**86.44**	**94.51**	94.20
Precision	87.69	**89.66**	94.34	**95.68**
IoU	76.91	**78.61**	89.44	**90.36**
F1	86.95	**88.02**	94.42	**94.93**

Table 4.8: Performance metrics of the state-of-the-art results and the models' predictions for the Inria and Whu datasets.

5

Conclusions and Future Work

This chapter presents the conclusions of the work developed, as well as the future work that can be done to improve the results obtained.

5.1 Conclusions

Creating of an automated system that accurately delineates building footprints from aerial images presents a critical yet challenging task. These footprints are the foundation for creating comprehensive maps essential for various applications, such as urban planning, disaster response, and infrastructure development. However, buildings' spatial variability and complexity introduce significant challenges to this task.

Using Deep Learning models has proven to be a promising approach for building footprint segmentation. However, the selection of architecture and datasets is crucial for the performance of the models. The primary objective of this dissertation was to develop a system capable of automatically delineating building footprints from aerial images. To achieve this objective, the process was divided into three main tasks: the creation of a new dataset for building footprint segmentation, the evaluation of two Deep Learning architectures, and a post-processing step to improve the obtained results.

DGT provided aerial images from a Portuguese region to create the new dataset. These images were used to construct two datasets, each containing 1085 images, and their corresponding ground truth masks. Taking advantage of the presence of the NIR band in the images, a false-color dataset was created and utilized for model evaluation. Since these datasets initially lacked associated building footprints, they were supplemented using the Microsoft *GlobalMLBuildingFootprints* dataset, which contains building footprint polygons in GeoJSON format from various locations worldwide. Two additional benchmark datasets, the Inria Aerial Labeling and the Whu Building datasets, were also employed for model evaluation.

During the model evaluation process, two Deep Learning architectures, ResUNet and SwinUNet, were compared for their effectiveness in building footprint segmentation. The results revealed that the ResUNet architecture consistently outperformed the SwinUNet

architecture across all evaluation metrics and datasets. Additionally, it became evident that the choice of datasets significantly influenced the models' performance. Specifically, the Whu dataset, characterized by lower variability in building shapes and sizes, yielded superior results to the Inria dataset, which exhibited higher variability in these aspects.

The DGT dataset, specifically created for this dissertation, provided a platform to investigate the impact of false-color images and imperfect footprints on model performance. It became evident that using imperfect footprints in the training process decreased model performance and created challenges in analyzing the results. Even if the model performs well, imperfections in the ground truth masks can impact the calculated metrics, as seen in the results obtained with the DGT dataset. A multi-data approach was used to mitigate this challenge, where the model was trained on different datasets to improve its generalization capabilities. Despite the low performance metrics obtained the model could generalize well to the DGT dataset. These results were confirmed by the visual inspection of the segmentation results, which showed that the model was able to accurately segment the building footprints, even with imperfections in the ground truth masks.

The post-processing step involved of applying a combination of spatial filtering and morphological operations to the segmentation masks, which helped refine the delineated building footprints. Specifically, a mean filter to remove minor imperfections, a dilation to fill gaps, and the Douglas–Peucker algorithm simplifies the building contours. The results obtained after post-processing reaffirmed the significance of these steps in enhancing the overall quality of the extracted building footprints.

The performance obtained by the system developed in this dissertation presents promising results, showing that Deep Learning models can be a good approach for building footprint segmentation. The system developed in this dissertation accurately delineate building footprints from aerial images fully automatedly and can be used in various fields. In addition, the advantage of using CNN-based models is confirmed, as they present the best performance in this task compared to Transformer-based methods. The use of hybrid-based methods, combining CNN-based models with Transformer-based models, was not employed in this dissertation since no computational resources were available to perform this task.

5.2 Future Work

Despite the promising results obtained in this dissertation, several aspects can still be improved and further explored. Some of the approaches that can be studied in future work to improve the results include:

- **Performance of the models using perfected masks:** Investigating the impact of using perfected masks in the training process could provide insights into the models' performance. It would allow for a better evaluation of the models' performance since the presence of imperfections in the ground truth masks can impact the calculated

metrics. Additionally, perfected masks could help mitigate the challenges posed by imperfect footprints in the ground truth masks and improve the results obtained by the models.

- **Exploration of other datasets:** Investigating the use of additional datasets for building footprint segmentation could enhance the models' performance. Datasets with diverse characteristics and challenges can be utilized to train the models, thereby improving their generalization capabilities.
- **Exploration of alternative architectures:** Exploring other architectures for building footprint segmentation could lead to improved results. Different architectures can capture varying patterns and features from the data, potentially enhancing performance in this task. Additionally, investigating hybrid-based methods that combine CNN-based models with Transformer-based models may offer further performance improvements by leveraging the strengths of both architectures.
- **Exploration of alternative post-processing techniques:** Exploring alternative post-processing techniques could improve the segmentation results. Different techniques can refine the results obtained by the models, resulting in more accurate delineations of building footprints.

Bibliography

[1] *Population: Demographic Situation, Languages and Religions*. URL: https://eurydice.eacea.ec.europa.eu/national-education-systems/portugal/population-demographic-situation-languages-and-religions (cit. on p. 1).

[2] Observatório Ordenamento do Território e Urbanismo. *Uso e Ocupação do Solo em Portugal Continental*. Flyer. URL: https://www.dgterritorio.gov.pt/download/folheto/folheto_cos_lq.pdf. 2020 (cit. on pp. 1, 2).

[3] W. Jochem and A. Tatem. "Tools for mapping multi-scale settlement patterns of building footprints: An introduction to the R package foot". In: *PLOS ONE* 16 (2021-02), e0247535. DOI: 10.1371/journal.pone.0247535 (cit. on p. 1).

[4] Z. Wang et al. "Urban building extraction from high-resolution remote sensing imagery based on multi-scale recurrent conditional generative adversarial network". In: *GIScience & Remote Sensing* 59.1 (2022), pp. 861–884. DOI: 10.1080/15481603.2022.2076382 (cit. on p. 1).

[5] J. Li et al. "A review of building detection from very high resolution optical remote sensing images". In: *GIScience & Remote Sensing* 59.1 (2022), pp. 1199–1225. DOI: 10.1080/15481603.2022.2101727 (cit. on p. 1).

[6] S. D. Khan, L. Alarabi, and S. Basalamah. "An Encoder–Decoder Deep Learning Framework for Building Footprints Extraction from Aerial Imagery". In: *Arabian Journal for Science and Engineering* 48.2 (2023-02), pp. 1273–1284. ISSN: 2191-4281. DOI: 10.1007/s13369-022-06768-8. URL: https://doi.org/10.1007/s13369-022-06768-8 (cit. on p. 1).

[7] Y. Shi, Q. Li, and X. X. Zhu. "Building Footprint Generation Using Improved Generative Adversarial Networks". In: *IEEE Geoscience and Remote Sensing Letters* 16.4 (2019), pp. 603–607. DOI: 10.1109/LGRS.2018.2878486 (cit. on p. 1).

[8] H. Thisanke et al. "Semantic segmentation using Vision Transformers: A survey". In: *Engineering Applications of Artificial Intelligence* 126 (2023), p. 106669. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.106669. URL: https:

//www.sciencedirect.com/science/article/pii/S0952197623008539 (cit. on p. 1).

[9] S. Yang et al. "Boundary-guided DCNN for building extraction from high-resolution remote sensing images". In: *The International Journal of Advanced Manufacturing Technology* (2022-05). DOI: 10.1007/s00170-022-09242-9 (cit. on p. 1).

[10] H. Guo et al. "A coarse-to-fine boundary refinement network for building footprint extraction from remote sensing imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 183 (2022), pp. 240–252. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2021.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0924271621002975 (cit. on pp. 2, 20).

[11] S. Minaee et al. "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2022), pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968 (cit. on pp. 2, 4).

[12] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on pp. 2, 4, 6, 7).

[13] R. Yamashita, M. Nishio, R. Do, et al. "Convolutional neural networks: an overview and application in radiology". In: *Insights Imaging* 9 (2018), pp. 611–629. DOI: 10.1007/s13244-018-0639-9 (cit. on pp. 2, 9, 10).

[14] C. Janiesch, P. Zschech, and K. Heinrich. "Machine learning and deep learning". In: *Electronic Markets* 31.3 (2021), pp. 685–695. ISSN: 1422-8890. DOI: 10.1007/s12525-021-00475-2. URL: https://doi.org/10.1007/s12525-021-00475-2 (cit. on p. 5).

[15] A. Jain, J. Mao, and K. Mohiuddin. "Artificial neural networks: a tutorial". In: *Computer* 29.3 (1996), pp. 31–44. DOI: 10.1109/2.485891 (cit. on p. 5).

[16] A. D. Rasamoelina, F. Adjailia, and P. Sinčák. "A Review of Activation Function for Artificial Neural Network". In: *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*. 2020, pp. 281–286. DOI: 10.1109/SAMI48414.2020.9108717 (cit. on p. 5).

[17] P. P. Shinde and S. Shah. "A Review of Machine Learning and Deep Learning Applications". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857 (cit. on p. 6).

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495. DOI: 10.1109/TPAMI.2016.2644615 (cit. on p. 6).

[19] K. Maharana, S. Mondal, and B. Nemade. "A review: Data pre-processing and data augmentation techniques". In: *Global Transitions Proceedings* 3.1 (2022). International Conference on Intelligent Engineering Approach(ICIEA-2022), pp. 91–99. ISSN: 2666-285X. DOI: https://doi.org/10.1016/j.gltp.2022.04.020. URL: https://www.sciencedirect.com/science/article/pii/S2666285X22000565 (cit. on p. 7).

[20] L. Perez and J. Wang. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. 2017. arXiv: 1712.04621 [cs.CV] (cit. on p. 7).

[21] K. Cho et al. "A Performance Comparison of Loss Functions". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 2019, pp. 1146–1151. DOI: 10.1109/ICTC46691.2019.8939902 (cit. on p. 7).

[22] Shaloni et al. "Building Extraction from Remote Sensing Images: A Survey". In: *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)* (2020), pp. 966–971. URL: https://api.semanticscholar.org/CorpusID:232149608 (cit. on p. 7).

[23] P. Borba et al. "Building Footprint Extraction Using Deep Learning Semantic Segmentation Techniques: Experiments and Results". In: 2021-07, pp. 4708–4711. DOI: 10.1109/IGARSS47720.2021.9553855 (cit. on p. 8).

[24] J. Terven et al. *Loss Functions and Metrics in Deep Learning*. 2023. arXiv: 2307.02694 [cs.LG] (cit. on p. 8).

[25] E. Fernandez-Moral et al. "A New Metric for Evaluating Semantic Segmentation: Leveraging Global and Contour Accuracy". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1051–1056. DOI: 10.1109/IVS.2018.8500497 (cit. on p. 8).

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 9, 10).

[27] A. Zhang et al. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023 (cit. on p. 9).

[28] Z. Li et al. "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (2022), pp. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827 (cit. on p. 9).

[29] W. Rawat and Z. Wang. "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review". In: *Neural Computation* 29.9 (2017), pp. 2352–2449. DOI: 10.1162/neco_a_00990 (cit. on p. 10).

[30] N. Aloysius and M. Geetha. "A review on deep convolutional neural networks". In: *2017 International Conference on Communication and Signal Processing (ICCSP)*. 2017, pp. 0588–0592. DOI: 10.1109/ICCSP.2017.8286426 (cit. on pp. 10, 11).

[31] L. Alzubaidi, J. Zhang, A. Humaidi, et al. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: *J Big Data* 8 (2021), p. 53. DOI: 10.1186/s40537-021-00444-8 (cit. on p. 11).

[32] S. Jamil, M. Jalil Piran, and O.-J. Kwon. "A Comprehensive Survey of Transformers for Computer Vision". In: *Drones* 7.5 (2023). ISSN: 2504-446X. DOI: 10.3390/drones7050287. URL: https://www.mdpi.com/2504-446X/7/5/287 (cit. on pp. 11, 12).

[33] Y. Liu et al. "A Survey of Visual Transformers". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–21. DOI: 10.1109/TNNLS.2022.3227717 (cit. on pp. 12, 13, 18).

[34] A. A. Aleissaee et al. "Transformers in Remote Sensing: A Survey". In: *Remote Sensing* 15.7 (2023). ISSN: 2072-4292. DOI: 10.3390/rs15071860. URL: https://www.mdpi.com/2072-4292/15/7/1860 (cit. on pp. 12, 13).

[35] P. Mehrani and J. K. Tsotsos. *Self-attention in Vision Transformers Performs Perceptual Grouping, Not Attention*. 2023. arXiv: 2303.01542 [cs.CV] (cit. on p. 13).

[36] B. Bischke et al. "Multi-Task Learning for Segmentation of Building Footprints with Deep Neural Networks". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1480–1484. DOI: 10.1109/ICIP.2019.8803050 (cit. on p. 14).

[37] C. Chawda, J. Aghav, and S. Udar. "Extracting Building Footprints from Satellite Images using Convolutional Neural Networks". In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2018, pp. 572–577. DOI: 10.1109/ICACCI.2018.8554893 (cit. on p. 14).

[38] D. Chudasama et al. "Image Segmentation using Morphological Operations". In: *International Journal of Computer Applications* 117 (2015-05), pp. 16–19. DOI: 10.5120/20654-3197 (cit. on pp. 14, 15).

[39] R. Gonzalez and R. Woods. *Digital Image Processing Global Edition*. Pearson Deutschland, 2017, p. 1024. ISBN: 9781292223049. URL: https://elibrary.pearson.de/book/99.150005/9781292223070 (cit. on pp. 14–16).

[40] P. Li et al. "Overview of Image Smoothing Algorithms". In: *Journal of Physics: Conference Series* 1883 (2021-04), p. 012024. DOI: 10.1088/1742-6596/1883/1/012024 (cit. on pp. 15, 16).

[41] E. Maggiori et al. "Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark". In: *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017, pp. 3226–3229. DOI: 10.1109/IGARSS.2017.8127684 (cit. on p. 17).

[42] J. Liu and S. Ji. "A Novel Recurrent Encoder-Decoder Structure for Large-Scale Multi-View Stereo Reconstruction From an Open Aerial Dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020-06 (cit. on p. 17).

[43] V. Mnih. "Machine Learning for Aerial Image Labeling". PhD thesis. University of Toronto, 2013 (cit. on p. 17).

[44] G. Cheng et al. "Remote Sensing Image Scene Classification Meets Deep Learning: Challenges, Methods, Benchmarks, and Opportunities". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 3735–3756. ISSN: 2151-1535. DOI: 10.1109/jstars.2020.3005403. URL: http://dx.doi.org/10.1109/JSTARS.2020.3005403 (cit. on p. 18).

[45] M. Z. Alom et al. *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. 2018. arXiv: 1803.01164 [cs.CV] (cit. on p. 18).

[46] Z. Dong et al. "Distilling Segmenters From CNNs and Transformers for Remote Sensing Images' Semantic Segmentation". In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–14. DOI: 10.1109/TGRS.2023.3290411 (cit. on pp. 18, 19).

[47] F. Yuan, Z. Zhang, and Z. Fang. "An effective CNN and Transformer complementary network for medical image segmentation". In: *Pattern Recognition* 136 (2023), p. 109228. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.109228. URL: https://www.sciencedirect.com/science/article/pii/S0031320322007075 (cit. on pp. 18, 19).

[48] R. Pascanu, T. Mikolov, and Y. Bengio. *On the difficulty of training Recurrent Neural Networks*. 2013. arXiv: 1211.5063 [cs.LG] (cit. on p. 18).

[49] L.-C. Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2018), pp. 834–848. DOI: 10.1109/TPAMI.2017.2699184 (cit. on p. 18).

[50] H. Zhao et al. "Pyramid Scene Parsing Network". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6230–6239. DOI: 10.1109/CVPR.2017.660 (cit. on p. 18).

[51] J. Ni et al. "DNL-Net: Deformed Non-Local Neural Network for Blood Vessel Segmentation". In: *BMC Medical Imaging* 22.1 (2022-06), p. 109. ISSN: 1471-2342. DOI: 10.1186/s12880-022-00836-z. URL: https://doi.org/10.1186/s12880-022-00836-z (cit. on p. 18).

[52] H. Wu et al. "Optimized HRNet for image semantic segmentation". In: *Expert Systems with Applications* 174 (2021), p. 114532. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114532. URL: https://www.sciencedirect.com/science/article/pii/S0957417420311763 (cit. on p. 18).

[53] K. Sun et al. *Deep High-Resolution Representation Learning for Human Pose Estimation*. 2019. arXiv: 1902.09212 [cs.CV] (cit. on p. 18).

[54] A. Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV] (cit. on p. 19).

[55] Z. Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV] (cit. on p. 19).

[56] W. Wang et al. "PVT v2: Improved baselines with Pyramid Vision Transformer". In: *Computational Visual Media* 8.3 (2022-03), pp. 415–424. DOI: 10.1007/s41095-022-0274-8. URL: https://doi.org/10.1007%2Fs41095-022-0274-8 (cit. on p. 19).

[57] S. Zheng et al. *Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers*. 2021. arXiv: 2012.15840 [cs.CV] (cit. on p. 19).

[58] Y. Zhang, H. Liu, and Q. Hu. *TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation*. 2021. arXiv: 2102.08005 [cs.CV] (cit. on p. 19).

[59] L. Xu et al. "BCTNet: Bi-Branch Cross-Fusion Transformer for Building Footprint Extraction". In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–14. DOI: 10.1109/TGRS.2023.3262967 (cit. on p. 19).

[60] F. Yuan, Z. Zhang, and Z. Fang. "An effective CNN and Transformer complementary network for medical image segmentation". In: *Pattern Recognition* 136 (2023), p. 109228. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2022.109228. URL: https://www.sciencedirect.com/science/article/pii/S0031320322007075 (cit. on p. 19).

[61] Z. Dong et al. "Distilling Segmenters From CNNs and Transformers for Remote Sensing Images' Semantic Segmentation". In: *IEEE Transactions on Geoscience and Remote Sensing* 61 (2023), pp. 1–14. DOI: 10.1109/TGRS.2023.3290411 (cit. on p. 19).

[62] J. Wang et al. "DualSeg: Fusing transformer and CNN structure for image segmentation in complex vineyard environment". In: *Computers and Electronics in Agriculture* 206 (2023), p. 107682. ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2023.107682. URL: https://www.sciencedirect.com/science/article/pii/S0168169923000704 (cit. on p. 19).

[63] Y. Zhang, H. Liu, and Q. Hu. *TransFuse: Fusing Transformers and CNNs for Medical Image Segmentation*. 2021. arXiv: 2102.08005 [cs.CV] (cit. on p. 19).

[64] R. Zhang et al. "DSAT-Net: Dual Spatial Attention Transformer for Building Extraction From Aerial Images". In: *IEEE Geoscience and Remote Sensing Letters* 20 (2023), pp. 1–5. DOI: 10.1109/LGRS.2023.3304377 (cit. on p. 20).

[65] L. Wang et al. "Building Extraction With Vision Transformer". In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–11. ISSN: 1558-0644. DOI: 10.1109/tgrs.2022.3186634. URL: http://dx.doi.org/10.1109/TGRS.2022.3186634 (cit. on p. 20).

[66] S. K. McFEETERS. "The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features". In: *International Journal of Remote Sensing* 17.7 (1996), pp. 1425–1432. DOI: 10.1080/01431169608948714. URL: https://doi.org/10.1080/01431169608948714 (cit. on p. 24).

[67] *GlobalMLBuildingFootprints*. https://github.com/microsoft/GlobalMLBuildingFootprints/tree/main. Accessed: [Access Date] (cit. on p. 25).

[68] A. Toet and J. Walraven. "New false color mapping for image fusion". In: *Optical Engineering* 35 (1996-03). DOI: 10.1117/1.600657 (cit. on p. 26).

[69] E. Maggiori et al. "Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE. 2017 (cit. on p. 26).

[70] Wuhan University. *WHU Building Dataset*. http://gpcv.whu.edu.cn/data/building_dataset.html. Accessed: April 18, 2024 (cit. on pp. 27, 28).

[71] M.-J. KIM and J.-H. KIM. "DEVELOPMENT OF CONVOLUTIONAL NEURAL NETWORK MODEL FOR CLASSIFICATION OF CARDIOMEGALY X-RAY IMAGES". In: *Journal of Mechanics in Medicine and Biology* 22.08 (2022), p. 2240020. DOI: 10.1142/S0219519422400206. eprint: https://doi.org/10.1142/S0219519422400206. URL: https://doi.org/10.1142/S0219519422400206 (cit. on p. 29).

[72] X. Hao et al. "A Review of Data Augmentation Methods of Remote Sensing Image Target Recognition". In: *Remote Sensing* 15.3 (2023). ISSN: 2072-4292. DOI: 10.3390/rs15030827. URL: https://www.mdpi.com/2072-4292/15/3/827 (cit. on p. 30).

[73] O. Adedeji et al. *Image Augmentation for Satellite Images*. 2022. arXiv: 2207.14580 [cs.CV] (cit. on p. 30).

[74] K. Alomar, H. I. Aysel, and X. Cai. "Data Augmentation in Classification and Segmentation: A Survey and New Strategies". In: *Journal of Imaging* 9.2 (2023). ISSN: 2313-433X. DOI: 10.3390/jimaging9020046. URL: https://www.mdpi.com/2313-433X/9/2/46 (cit. on p. 30).

[75] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (cit. on pp. 30, 31).

[76] Z. Zhang, Q. Liu, and Y. Wang. "Road Extraction by Deep Residual U-Net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018-05), pp. 749–753. ISSN: 1558-0571. DOI: 10.1109/lgrs.2018.2802944. URL: http://dx.doi.org/10.1109/LGRS.2018.2802944 (cit. on pp. 31, 32).

[77] *Segmentation Models's Documentation*. https://smp.readthedocs.io/en/latest/index.html. Accessed: April 18, 2024 (cit. on p. 31).

[78] H. Cao et al. *Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation*. 2021. arXiv: 2105.05537 [eess.IV] (cit. on pp. 31, 33).

[79] *PyTorch Lightning Documentation*. https://lightning.ai/docs/pytorch/stable/. Accessed: April 18, 2024 (cit. on p. 33).

[80] U. Ramer. "An iterative procedure for the polygonal approximation of plane curves". In: *Computer Graphics and Image Processing* 1.3 (1972), pp. 244–256. ISSN: 0146-664X. DOI: https://doi.org/10.1016/S0146-664X(72)80017-0. URL: https://www.sciencedirect.com/science/article/pii/S0146664X72800170 (cit. on p. 36).

I

Training and validation loss

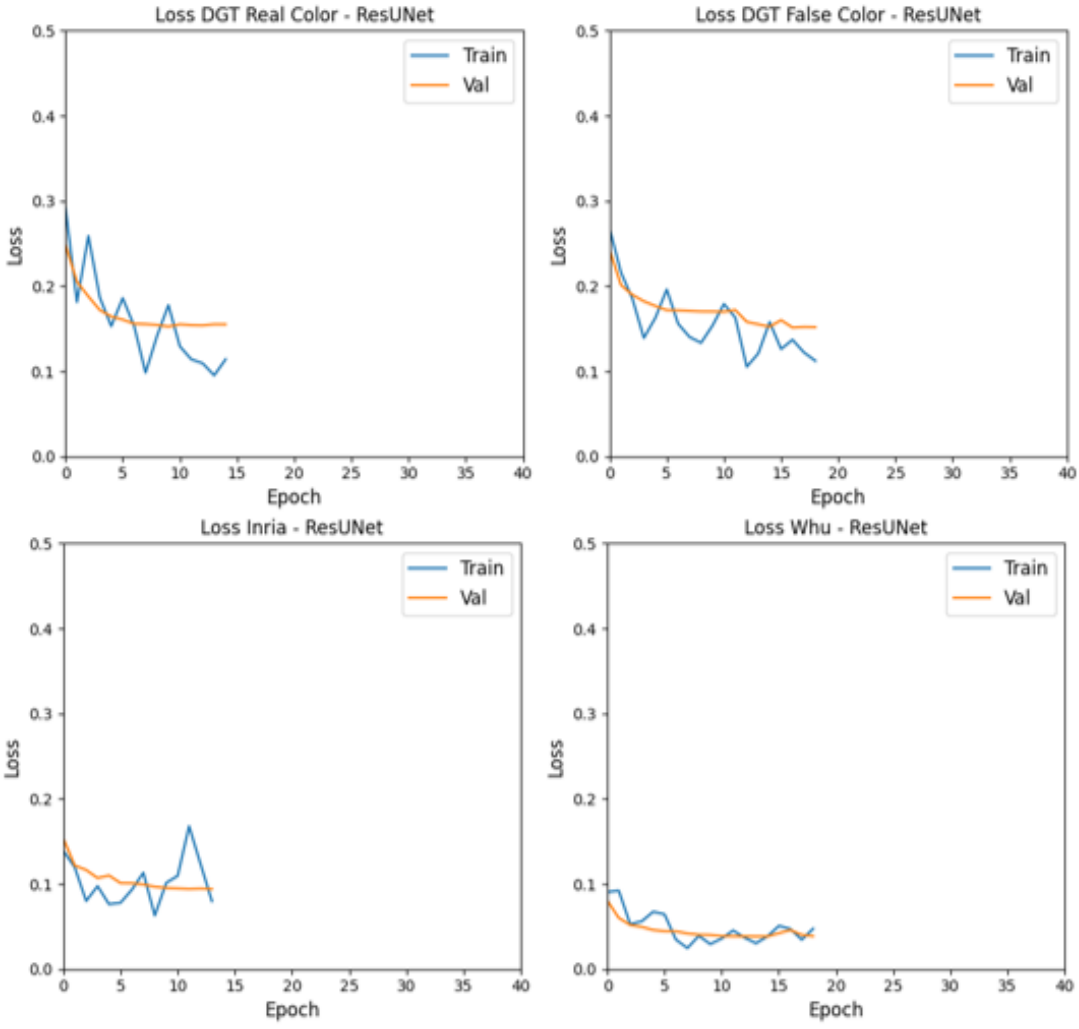


Figure I.1: Training and validation loss of ResUNet models in different datasets.

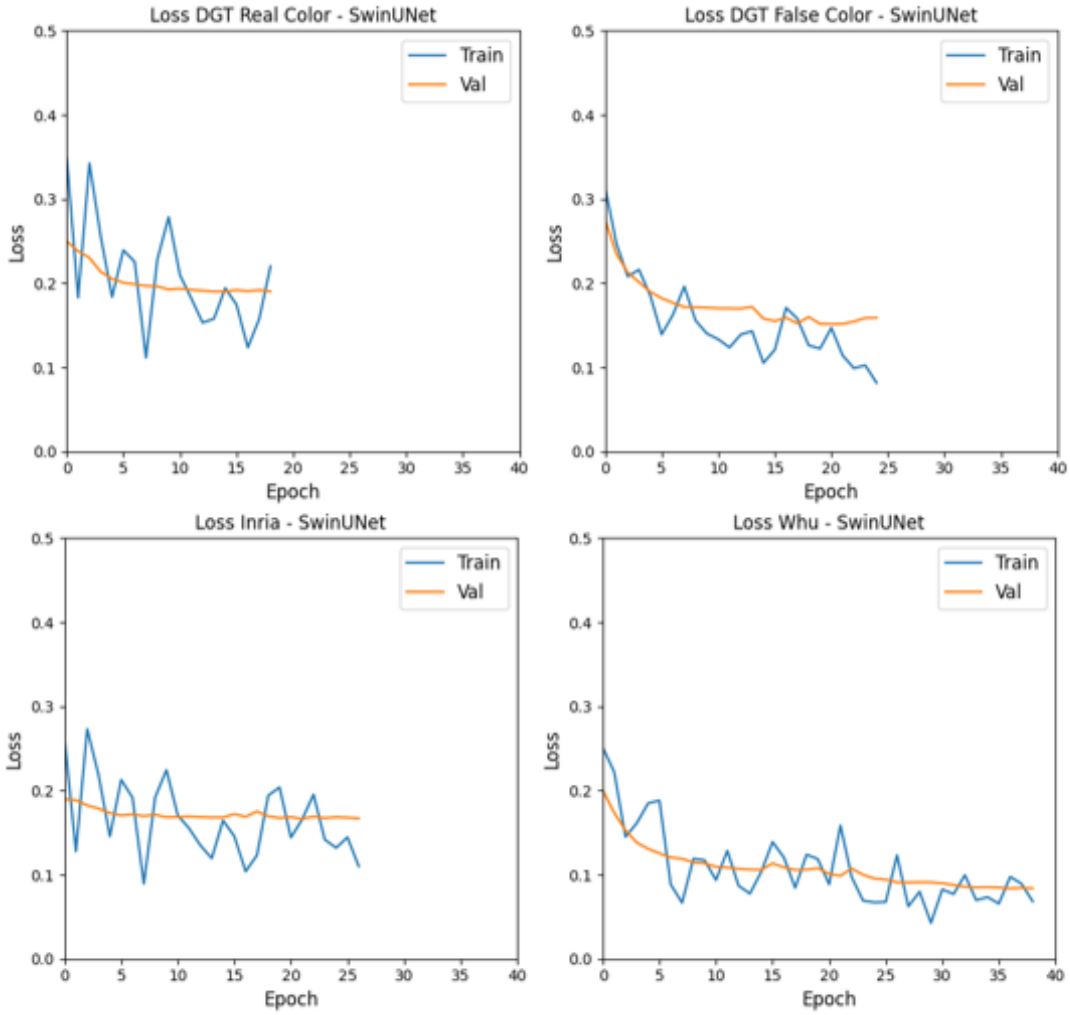


Figure I.2: Training and validation loss of SwinUNet models in different datasets.