People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University of Ain Temouchent - BELHADJ Bouchaib

Faculty of Science and Technology
Department of Mathematics and Computer Science

Hybrid CNN–Transformer Architecture

for Semantic Segmentation of High-Resolution Satellite Images

FINAL YEAR PROJECT

Speciality: Cybersecurity and Artificial Intelligence

Presented by:

Sarra MEZADA
Chaimaa Nesrine KRADEM CHARAH

Jury Members:

President:	Dr. BOUHALOUAN D.
Examiner:	Dr. BENOMAR M.L.
Supervisor:	Dr. MESSAOUDI M.A.

Academic Year: 2025-2026

Dedication

﴿وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ﴾

First and foremost, I praise and thank Allah Almighty for giving me the strength, health, and patience to complete this journey. Everything I achieve is through his grace and blessings.

To the pillars of my life, my beloved mother and father: This work is the fruit of your endless love, prayers, and sacrifices. Thank you for always standing by my side and supporting me through my best and my hardest days. I have reached this milestone by the grace of Allah and because of your deep faith in me.

To my wonderful brothers, sister, and dear aunt: Thank you for your constant encouragement, your smiles, and for always being my safe place.

To my beautiful partner: Thank you for walking this path beside me, for your unwavering patience, and for believing in me even when doubts crept in. Your presence has been my peace and motivation throughout this journey.

To my supervisor: I express my deepest gratitude for your invaluable guidance, immense patience, and academic support. Thank you for your trust, your constructive insights, and for helping us steer this project to success.

I dedicate this achievement to all of you.

Sarra MEZADA

Dedication

﴿قُلِ اعْمَلُوا۟ فَسَيَرَى ٱللَّهُ عَمَلَكُمْ وَرَسُولُهُۥ وَٱلْمُؤْمِنُونَ﴾

By the grace and guidance of Allah. The journey was not short, nor was the path paved with ease, and these final moments, after years devoted to knowledge and learning, now stand at the threshold of my graduation.

I dedicate this achievement first to the ambitious, hardworking child I once was, who persevered throughout her school years and continued the path to accomplish what she could be proud of ***to myself****.*

To the one who gave my name its honor, who was my support and my pride ***to my dear father****, my refuge and my strength, my pride and my first love.*

To the most precious person in my life, the secret behind my strength and success, who was my pillar and my power after God, my friend and my first support, I dedicate this success to you with all my pride and gratitude, and I ask God to protect you and keep you a light in my life ***my beautiful mother****.*

To my unwavering support and the safety of my days, to those whose presence has been a wellspring from which I drew peace .. my brother and sisters, ***Mohamed****,* ***Mokhtaria****, and* ***Zahra****.*

To the one who accompanied me through my very first university days, my friend ***Romaissa****, who shared with me the joy of beginnings and the hardship of the early days, and stood by my side at every step, I dedicate this success to you, for you were and still are a blessing I will never forget.*

And to ***Rahma****, a lifelong friend whose worth cannot be measured in words. And to my friend* ***Khadija****, who shared with me the most beautiful and the hardest moments, my companion throughout this university journey.*

To my partner in this journey, who was my support in the difficult moments and my ally in success ***Sarra****, thank you for your cooperation, dedication, and patience throughout this experience, and I wish you every success in your future path.*

*To my supervisor,****Dr. Messaoudi****, who generously shared his valuable advice and guidance throughout this work, I extend my sincerest thanks, appreciation, and respect. To every teacher and professor who helped shape our path toward knowledge.*

And finally, to everyone I have mentioned and everyone I have not: I dedicate this work to you, the harvest of years of effort and patience, for you were the best support and help in reaching it. Praise be to Allah, first and last, for the blessing of success.

Chaimaa Nesrine KRADEM CHARAH

Acknowledgements

First and foremost, we raise our praise to God Almighty, Master of destinies, for His countless blessings. His infinite mercy, and the strength He instilled in us throughout this journey. Without His grace, this thesis could not have been completed.

We express our deepest gratitude to our supervisor, Dr. MESSAOUDI A., for the wealth of his advice and the soundness of his guidance, as well as for his availability, patience, and exemplary dedication. His support was a true pillar throughout this academic journey.

Our thanks also go to all the teachers of the mathematics and computer science departments for the quality of their teaching and their dedication throughout our studies. We extend our deep gratitude to the members of the jury, Dr. BENOMAR M.L. and Dr. BOUHALOUAN D, for the honor they have given us by evaluating this work, as well as for their relevant and enriching feedback. May they find here the expression of our most sincere esteem.

Our most affectionate and grateful thoughts go to our families, true havens of support and unconditional love. Our sincere thanks also go to our colleagues and friends for their moral support, their availability, and the spirit of solidarity they have always shown.

Finally, our thanks go to everyone who, in one way or another, left their mark on this project through a gesture, a word, a glance, or quiet support.

Abstract

Semantic segmentation of high-resolution satellite imagery is a crucial task for applications such as urban planning and environmental monitoring. However modern satellite images are very complicated, containing objects of different sizes and irregular boundaries. Convolutional Neural Networks (CNNs) are powerful tools for extracting local details such as textures and edges but they struggle to capture long-range contextual information. On the other hand, Vision Transformers (ViTs) are good at context capturing on the global level but lack local precision and typically require massive amounts of training data.

The project is intended to address this gap by developing a hybrid CNN-Transformer architecture. The network combines the strengths of CNNs for local feature extraction with the power of Transformers for global context understanding, leading to improved segmentation performance, particularly for large objects and complex landscape classes.

The proposed hybrid model is designed, implemented, and tested on annotated satellite datasets, with performance measured using pixel accuracy and the Dice coefficient. The results show that the hybrid framework, especially with transfer learning, provides a significant improvement in edge detection and generalizes better in different environments. This work shows the importance of AI in remote sensing and offers a solid tool for automated decision-making systems.

Keywords: Semantic Segmentation, Satellite Imagery, High-Resolution, Deep Learning, Hybrid Architecture, CNN–Transformer, Remote Sensing, Transfer Learning.

Résumé

La segmentation sémantique d'images satellitaires haute résolution est une tâche cruciale pour des applications telles que l'aménagement urbain et la surveillance environnementale.

Cependant, les images satellitaires modernes sont très complexes et contiennent des objets de tailles variées et aux contours irréguliers.

Les réseaux de neurones convolutifs (CNN) sont des outils puissants pour extraire des détails locaux tels que les textures et les contours, mais ils peinent à capturer des informations contextuelles à longue portée.

D'autre part, les Vision Transformers (ViT) sont performants pour la capture du contexte à l'échelle globale, mais manquent de précision locale et nécessitent généralement une quantité massive de données d'entraînement. Ce projet vise à combler cette lacune en développant une architecture hybride CNN-Transformer. Le réseau combine les atouts des CNN pour l'extraction de caractéristiques locales avec la puissance des Transformers pour la compréhension du contexte global, ce qui permet d'améliorer les performances de segmentation, notamment pour les grands objets et les paysages complexes.

Le modèle hybride proposé est conçu, implémenté et testé sur des jeux de données satellitaires annotés.

Ses performances sont mesurées à l'aide de la précision au pixel et du coefficient de Dice. Les résultats montrent que le cadre hybride, notamment grâce à l'apprentissage par transfert, améliore significativement la détection des contours et se généralise mieux dans différents environnements. Ce travail démontre l'importance de l'IA en télédétection et offre un outil performant pour les systèmes de décision automatisés.

Mots-clés : Segmentation sémantique, Images satellites, Haute résolution, Deep Learning, Architecture hybride, CNN–Transformer, Télédétection, Apprentissage par transfert.

الملخص

يُعدّ التجزئة الدلالية لصور الأقمار الصناعية عالية الدقة مهمة بالغة الأهمية لتطبيقات مثل التخطيط الحضري والرصد البيئي. ومع ذلك، فإن صور الأقمار الصناعية الحديثة معقدة للغاية، إذ تحتوي على أجسام بأحجام مختلفة وحدود غير منتظمة. تُعتبر الشبكات العصبية الالتفافية (CNNs) أدوات فعّالة لاستخراج التفاصيل المحلية مثل الأنسجة والحواف، لكنها تواجه صعوبة في التقاط المعلومات السياقية بعيدة المدى. من ناحية أخرى، تتميز محولات الرؤية (ViTs) بقدرتها على التقاط السياق على المستوى العالمي، لكنها تفتقر إلى الدقة المحلية، وعادةً ما تتطلب كميات هائلة من بيانات التدريب.

يهدف هذا المشروع إلى معالجة هذه الفجوة من خلال تطوير بنية هجينة تجمع بين CNN وTransformer. تجمع هذه الشبكة بين نقاط قوة CNNs في استخراج الميزات المحلية وقوة Transformers في فهم السياق العالمي، مما يؤدي إلى تحسين أداء التجزئة، لا سيما بالنسبة للأجسام الكبيرة وفئات المناظر الطبيعية المعقدة. تم تصميم النموذج الهجين المقترح وتنفيذه واختباره على مجموعات بيانات الأقمار الصناعية المشروحة، مع قياس الأداء باستخدام دقة البكسل ومعامل دايس.

تُظهر النتائج أن الإطار الهجين، وخاصةً مع التعلم بالنقل، يُحسّن بشكل ملحوظ من اكتشاف الحواف ويُعمّم بشكل أفضل في بيئات مختلفة. يُبرز هذا العمل أهمية الذكاء الاصطناعي في الاستشعار عن بُعد، ويُقدّم أداةً فعّالة لأنظمة اتخاذ القرار الآلي.

الكلمات المفتاحية: التجزئة الدلالية، صور الأقمار الصناعية، الدقة العالية، التعلم العميق، البنية الهجينة، شبكة عصبية الالتفافية - محوّل، الاستشعار عن بُعد، التعلم بالنقل.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
ViT	Vision Transformer
FC	Fully Connected
NLP	Natural Language Processing
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
Adam	Adaptive Moment Estimation
RS	Remote Sensing
EMR	Electromagnetic Radiation
UV	Ultraviolet
IR	Infrared
NIR	Near-Infrared
SWIR	Shortwave Infrared
RGB	Red, Green, Blue
DN	Digital Number
GIS	Geographic Information System
SAR	Synthetic Aperture Radar
GEOBIA	Geographic Object-Based Image Analysis
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer

IoU	Intersection over Union
mIoU	Mean Intersection over Union
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
GPU	Graphics Processing Unit
VRAM	Video Random Access Memory
GUI	Graphical User Interface
LoveDA	Land-Cover Domain Adaptive

General Introduction

In the last few decades the Earth has undergone rapid environmental and structural changes due to the combined effects of human activity and natural phenomena. Accurate, large-scale monitoring is needed to address massive urbanization, industrial growth, climate change and changing ecosystems. In this context, the ability to observe, analyze and map these changes has become essential for scientific research and sustainable development. This data is of vital importance for public decision making in urban planning, resource management and environmental protection. To fill this need remote sensing has become a necessary tool. The new generation of satellite constellations can image large geographical areas with high resolution and at regular time intervals. These satellite images are rich in both spectral and spatial information.

However, the sheer volume and increasing complexity of next generation satellite data means that manual analysis or traditional image processing methods (like basic thresholding or edge detection) are no longer adequate. Automatic analysis of high resolution remote sensing data is a major goal in computer vision, where semantic segmentation is a crucial component. This problem consists of the assignment of a specific thematic label (e.g. building, water, vegetation, bare soil) to each individual pixel of a scene. Deep Learning (DL) has totally revolutionized this field during the last years. Traditional convolutional neural networks (CNNs), like U-Net, have shown great power in capturing local hierarchical features like sharp edges and fine textures.

Despite their success, CNNs are inherently limited by a small receptive field, preventing them from capturing long-range contextual relationships in large images. More recently, Vision Transformers (ViTs) have come to the scene and have shown very successful in modeling global context and understanding how different parts of an entire image relate to each other. Pure transformers, however, lack local inductive biases, struggle to achieve fine boundary precision, and require massive amounts of training data to generalize well. This project is related to this particular technological challenge. The first goal of this work is to design and develop a hybrid CNN–Transformer architecture for semantic segmentation of high-resolution satellite images.

Our proposed network aims to exploit the local feature-extraction capability of convolutional layers and the global context-modeling capability of Transformer blocks in a strategic way, thereby bridging the gap between fine-grained boundary detail and macro-level scene understanding. This is particularly important for

complex land-cover classes and large scale structures that require broad contextual awareness.

To accomplish this hybrid architecture, our development workflow was executed in cloud-based deep learning environment, namely Kaggle. We used the Python language with industry-standard frameworks, namely, TensorFlow and Keras for core modeling, NumPy for data processing, and Matplotlib for visualizing the data. We used free cloud GPU acceleration to handle the high computational demands of our network. Our methodology follows a hybrid stream modeling approach of specialized convolutional backbones and self-attention transformer blocks. To accelerate the convergence speed and solve the data scarcity problem, a transfer learning strategy with pre-trained weights was adopted. Finally, we performed a rigorous performance evaluation with standard computer vision metrics such as pixel accuracy, Dice coefficient and a detailed confusion matrix analysis to measure the robustness of the model.

This manuscript is structured in four complementary chapters to present this work in a comprehensive manner. Chapter 1 introduces the fundamentals of remote sensing, including Earth observation, its historical development, and the concepts of electromagnetic radiation and its spectrum. Chapter 2 builds on this by exploring satellite imagery and semantic segmentation, covering basic principles, essential preprocessing techniques, and various image segmentation approaches, from classical methods onwards.

The remainder of the chapters describes the theoretical core and practical validation of our development. Chapter 3 presents the theoretical background of deep learning. It goes through the standard convolutional mechanisms, the self-attention mechanism behind Vision Transformers and the prominent baseline architectures that inspired our hybrid framework.

Finally, Chapter 4 concludes this work by explaining the implementation, experiments, and results which focus on the core contribution of this thesis via explanation of the benchmark datasets employed, implementation process of the proposed hybrid model, and comparison of results with the baseline CNN and Transformer models. The manuscript is then concluded with an overall summary, highlighting the core contributions of this thesis, pointing out some practical limitations, and suggesting possible directions for future research.

Chapter 1

Related Works

Recent literature on semantic segmentation of high-resolution satellite images Underlines three main methodological approaches: convolutional neural networks, transformer-based models, and hybrid CNN–Transformer architectures.

The architectural foundation of modern computer vision was established by LeCun et al. (1989) at AT&T Bell Laboratories; they applied a backpropagation algorithm to recognize handwritten digits taken from the US Mail[1].

Taye presented a comprehensive theoretical review of CNNs, underlining that CNNs are able to automatically learn a hierarchy of features for image recognition and classification and the fundamental building blocks of CNNs, including the convolutional, pooling, and activation layers[2].

Since then, Bhatt et al. (2021) trace the historical development of CNNs, starting from early architectures such as LeNet and AlexNet and then to more advanced models such as VGG, ResNet, Inception, DenseNet, and other modern CNN variants to improve accuracy and efficiency[3].

Another study, Performed by Salim et al., was a comparative analysis of modern CNN architectures, evaluating their performance. Over different datasets to identify the strengths and weaknesses of models such as AlexNet, VGG, and ResNet. They have noted that the primary limitation of CNN architectures is their restricted receiving field, which focuses on the neighborhoods adjacent to local pixels and often fails to capture the global context and long-range dependencies. Contained in image[4].

To target the restricted receptive field of traditional CNNs, the field of remote sensing has recently examined transformer-based models.

As discussed by Patwardhan et al. (2023), Transformers have fundamentally transformed Natural Language Processing (NLP) by replacing traditional recurrent models using a mechanism that can processes data sequences in parallel.

This success is primarily assigned to the self-attention mechanism, which allows the model to learn relationships between all elements in a sequence in parallel, without regard to their position[5].

Dosovitskiy et al. have demonstrated that unlike CNNs, which are limited by local inductive biases, the Vision Transformer (ViT) can learn global relationships

across the entire image from the first layers, giving us more comprehensive spatial representation for complex recognition tasks[6].

As indicated in the recent survey by Aleissaee et al. (2023), Transformers utilize a self-attention mechanism to capture long-range dependencies[7].

Jakubik et al. (2023) recently introduced a geospatial foundation model that marks a shift away from task-specific architectures, by pre-training a Vision Transformer on a massive unlabeled datasets from the Harmonized Landsat-Sentinel 2 (HLS) archive, they have demonstrated that a single generalist model can be efficiently fine-tuned for a wide range of earth observation tasks, Such as flood mapping and wildfire segmentation[8].

Recent research has exploited the benefits of both architectures CNNs and Transformer.

Then, Wang et al. (2023) have addressed the limitations of Vision Transformers (ViT) by introducing P²FEVIT, a hybrid model that embeds plug-and-play CNN features into the ViT backbone, by fusing local convolutional features with global attention mechanisms, the architecture gives a very high-performance classification and reducing the heavy reliance on the large-scale pre-training data[9].

Zhang et al. (2023) have also developed a hybrid model that leverages the strengths of CNNs and Transformers for high-resolution water body extraction, by using a U-Net backbone for initial multi-scale feature extraction followed by a multi-scale Transformer module, this architecture has captured both fine-grained local textures and broad contextual semantic associations.

In this research, They have addresses the boundary inaccuracies and poor continuity issues prevalent in traditional convolutional networks[10].

Chen et al. (2023) argued that for the semantic segmentation of remote sensing images, a fusion approach is superior to unimodal architectures, they have demonstrated that while CNNs can preserve fine spatial details, Transformers can provide the global semantic consistency that is to classify complex urban scenes[11].

To further enhance the performance of hybrid architectures, Wang et al. (2024) highlighted that hybrid models use CNNs for extracting local features and Transformers for global-context modeling,Which has demonstrated its Effectiveness across tasks such as semantic segmentation[12].

Wu et al. (2024) proposed the SCDUNet++ model, which effectively integrates CNN and Transformer branches to move beyond the localized constraints of traditional convolutional networks. Their research has demonstrated that this hybrid configuration is very good to be used at extracting and fusing spectral and topographic information, It led to better differentiation between Landslides and similar earth features topography[13].

The effectiveness of this hybrid fusion was validated by multiple standard benchmarks, including HRRSD and DIOR, where the DConvTrans-LKA model achieved superior mAP results compared to traditional detectors like Faster R-CNN and pure Swin Transformers. This performance demonstrates that the integrating of the convolutional inductive biases with the Transformer-based global reasoning is the most effective strategy used currently in high-resolution remote sensing tasks[14].

To optimize the synergy between different feature styles, Yao et al. (2024) proposed SSNet, which utilizes a specialized 'Feature Inject Module' (FIM). Unlike

standard fusion techniques, this module condenses global information from a Transformer branch and injects it into a CNN branch. This process, enhanced by depthwise strip convolutions allowing the network to maintain a broad global field of view while significantly it improves the segmentation accuracy of tiny and complex objects in many remote sensing tasks[15].

Li et al. (2025) introduced CTHNet as a solution for landslide identification in fragmented terrains, exploiting the strengths of CNNs and Transformers. The architecture's hybrid encoder utilizes convolutional layers to preserve fine-grained spatial details and employing a transformer-based module to establish a global receptive field. This hybrid strategy ensures that the model can resolve delicate boundary details without missing the overall view that is essential for intelligent disaster mapping on the Loess Plateau[16].

He et al. (2025) introduced GOFENet, a hybrid Transformer-CNN architecture that strengthens the integration of GEOBIA-based object priors into the segmentation process, by using an Object-Aware Feature Fusion (OAFF) module. The model connects the gap between pixel-level deep learning and object-level geographic analysis. This hybrid approach effectively suppresses the 'salt-and-pepper' noise typical of remote sensing classification, thereby enhancing the spatial consistency of the final output.[17].

In summary, the transition from purely convolutional networks to hybrid CNN-Transformer architectures represents a paradigm shift in remote sensing. By integrating the inductive biases of CNNs for spatial detail with the global reasoning of Transformers, current models can finally resolve the 'scale-variation' and 'contextual dependency' issues inherent in high-resolution satellite imagery. This development forms the direct motivation for the hybrid architecture proposed in this work.

Table 1.1: Summary of Methodological Evolution in Satellite Image Segmentation

Category	**Key Architectures**	**Strengths & Limitations**
Traditional CNNs (LeCun, ResNet, VGG)	LeNet, AlexNet, VGG, ResNet, DenseNet	**Pros:** Excellent local feature extraction (edges, textures). **Cons:** Restricted receptive field; fails to capture global context.
Pure Transformers (Dosovitskiy, etc.)	ViT, Prithvi (Geospatial Foundation Model)	**Pros:** Global relationships and long-range dependencies via self-attention. **Cons:** High data dependency; loses fine spatial details.
Hybrid CNN-Transformer (Current State)	TransUNet, P²FEVIT, SCDUNet++, CTHNet	**Pros:** Fuses local convolutional biases with global reasoning; superior boundary accuracy. **Cons:** Increased architectural complexity.

Chapter 2

Remote Sensing

2.1 Introduction

Remote sensing plays a crucial role in observing and understanding the Earth's surface and atmosphere without direct contact. It is an emerging discipline that bridges the gap between various scientific and technological disciplines. Because its multidisciplinary design requires its practitioners to possess a good foundational knowledge in areas like physics, geology, and engineering, with the encouragement of ongoing collaboration across a wide range of research specialties.[18]. With the recent advances in sensor technology and data processing techniques, remote sensing data has become highly detailed and complex. These data need an efficient analysis methods to extract meaningful information, making remote sensing an important foundation for modern image processing and machine learning approaches. This chapter presents the fundamental concepts of remote sensing, including its historical evolutionits, principles, data types, system components, and common applications.

2.2 Remote Sensing definition

Remote sensing is commonly characterized as both a science and a technical art, dedicated to extracting and analyzing information regarding Earth's features. This is achieved through the use of non contact sensors[19]. Its focuses on observing and measuring the physical and biological characteristics of the Earth's surface, oceans, and atmosphere using sensors mounted on satellites, aircraft, or other airborne platforms. Modern remote sensing systems generate digital data that can be processed and analyzed automatically, and after applying radiometric and geometric corrections, these data can be effectively integrated into Geographic Information Systems (GIS) for mapping and spatial analysis.

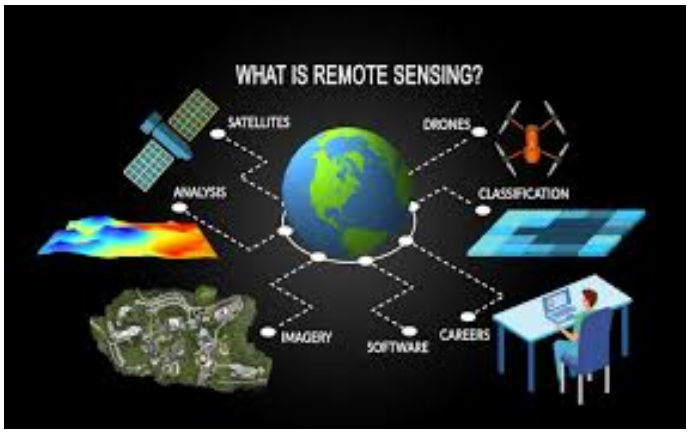


Figure 2.1: Remote sensing

Source: https://share.google/npjVZ5jq9ZvUmFwUk

2.3 Historical Evolution of Remote Sensing

The evolution of remote sensing can be categorized into five transformative eras, every one of them was distinguished by different modifications in technology and sensor capabilities.

1. **The Pioneering Era (1856 - WWI):** Initiated by Gaspard-Felix Turnachon (Nadar), who took the first aerial photograph from a hot air balloon in 1858, this era laid the foundations of photogrammetry and the reconstruction of three-dimensional terrain.

2. **Operational Aerial Photography (WWI - 1957):** Due to military necessity, aerial reconnaissance evolved into a strategic tool. The development of false-color infrared film was a major milestone, as it allowed for the first time the differentiation between healthy vegetation and camouflage.

3. **The Space Age (1957 - 1972):** After the launch of Sputnik, Earth observation moved to orbital platforms. Early meteorological satellites (ESSA) proved that global-scale monitoring was possible.

4. **The Landsat Revolution (1972 - 1990s):** The launch of Landsat 1 (formerly known as ERTS-1) introduced systematic, multispectral digital imaging. This shift from photo-interpretation to quantitative digital analys is the precursor to modern computer vision.

5. **Democratization and the AI Era (Present):** Today, spatial resolutions reach 30 cm (VHR).The rise of the Sentinel missions and companies like Maxar and Planet has brought us into the 'Big Data' environment. By combining this volume of data with high-performance computing, we can use Deep Learning for real-time global monitoring.

[20]

2.4 Remote Sensing Steps (Workflow)

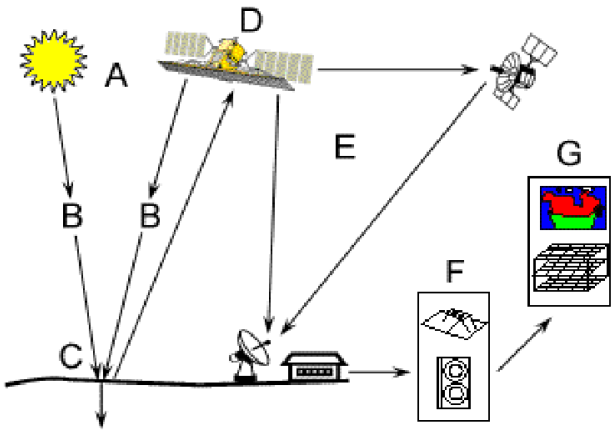


Figure 2.2: Remote sensing steps

Source: https://share.google/o8k3vJHO8ZUtUVXfg

The remote sensing workflow consists of seven main stages ,described as follows:

2.4.1 Energy Source or Illumination (A)

emits electromagnetic radiation directed toward the target of interest[21]. In passive remote sensing systems, the Sun acts as the natural source of electromagnetic radiation. otherwise in active remote sensing systems, such as radar or LiDAR, the sensor itself emits energy and measures the reflected signal.

2.4.2 Radiation and the Atmosphere (B)

The energy interacts with the intervening environment by passing from the source toward the target.[22]. This interaction includes scattering and absorption by atmospheric particles and gases, which may alter the signal before it reaches the target or the sensor.

2.4.3 Interaction with the Target (C)

Once the energy traverses the atmosphere, the interaction with the target is a function of both the intended subject and the electromagnetic radiation[23], creating an unique spectral signatures.

2.4.4 Recording of Energy by the Sensor (D)

The sensors integrated into satellites or airborne platforms detect and record the electromagnetic energy reflected or emitted from the target [24].

2.4.5 Transmission, Reception, and Processing (E)

Once recorded, the information is transmitted electronically to the ground receiving stations. [25], this involves processing steps such as radiographic and geometric corrections. This processing transforms the raw data into interpretable images.

2.4.6 Interpretation and Analysis (F)

The final imagery is interpreted through a combination of visual methods and digital or electronic analysis[26]. This analysis can be performed through visual interpretation or automated methods, including image classification, machine learning, and deep learning techniques.

2.4.7 Application (G)

In the final stage, the extracted information is applied to real-world problems such as forestry[27], Water Quality Parameters[28], agriculture [29], and urban planning[30]...

2.5 The Fundamental Components of a Remote Sensing System

2.5.1 The Energy Source (The Illuminator)

Remote sensing requires a source of electromagnetic energy to ”light up” the target. This energy is categorized based on its origin:

- **Passive Sensing:** Passive sensors capture the electromagnetic energy radiated from the Earth.which depends on surface properties and temperature [31]. They are commonly used to observe surface temperature, vegetation, oceans, and atmospheric conditions, but they are usually depend on sunlight.
- **Active Sensing:**In active remote sensing, the sensor propagates an artificial signal toward the target and then measures the return signal[31]. This method works day and night and is widely used for applications like terrain mapping, object detection, and environmental monitoring.

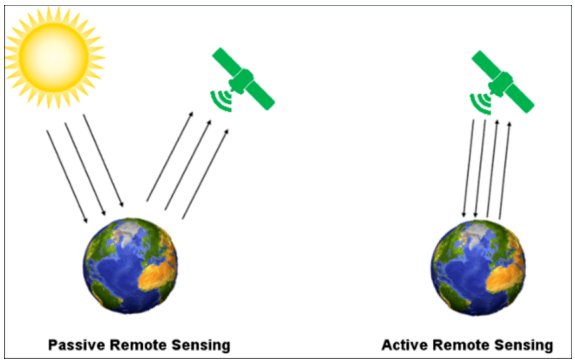


Figure 2.3: Differences between passive and active sensors

Source: https://share.google/CAVMQc4nakzQsdEIh

2.5.2 The Platform (The Vehicle)

Remote sensing platforms are the essential carriers for sensors and other components necessary for data acquisition[32].

Figure 2.4: Remote sensing platforms

Source: https://share.google/ksomVtrdph7azxZlT

- **Ground-based platforms:** UAVs are specialized aerial platforms that operates operate within a height range of 10 m to 150 m above ground level[32].
- **Airborne platforms:** This category includes helicopters, jet planes and airplanes, operating with an altitude range of 0.3 to 12 km [32].
- **Spaceborne platforms:** In spaceborne platforms, sensors are integrated into satellites orbiting the earth. These systems enable the acquisition of global imagery of earth, typically operating at altitudes ranging from 185 to 900 km[32].

2.5.3 The Sensor (The Eye)

Sensors are the device that is designed to receive electromagnetic radiations and converts these energy into signal , thes signals can be recorded or displayed as either

discrete values or an visual representation[32]. Different types of sensors are used in remote sensing applications:

- **Microwave Radiometer**
  A microwave radiometer is a passive remote sensing sensor that measures naturally emitted microwave radiation from the Earth. When mounted on satellites, it is considered a spaceborne (spatial) sensor and is widely used in remote sensing applications such as atmospheric sounding, ocean observation, soil moisture estimation, and sea surface temperature retrieval.

- **Gravimeter**
  A gravimeter is a remote sensing instrument used to measure variations in the Earth's gravitational field. When deployed on satellites, it is classified as a spaceborne sensor and is mainly used to study mass distribution, geophysical processes, ocean circulation, ice mass changes, and groundwater variations.

- **Optical Camera**
  An optical camera is a remote sensing instrument that captures reflected sunlight in the visible and near-infrared spectral bands to create images of the Earth's surface. When mounted on satellites, it is considered a spaceborne sensor and is widely used for land-use mapping, vegetation monitoring, urban planning, and environmental studies.

- **LiDAR**
  A LiDAR (Light Detection and Ranging) sensor is an active remote sensing instrument that emits laser pulses toward the Earth's surface quantifies the time-of-flight (ToF) required for the backscattered signal to return to the receiver. This allows it to accurately determine distances and generate detailed 3D elevation maps of terrain, vegetation, and urban structures. When installed on satellites, LiDAR is considered a spaceborne sensor.

2.5.4 The Target (The Subject)

The target is the specific feature or area on the Earth's surface such as: a forest, a city, or a body of water. We study these targets to learn more about them.

2.5.5 The Atmosphere

The atmosphere acts as a medium between the target we are looking at and the sensor used for capturing the information from this target. It can change the energy as it passes through , which sometimes creates "noise" in the data. this noise can be: Dust or smoke in the air and as a result it reduce image clarity , or can be air pollution causing distortion in urban images.

2.6 Application of Remote Sensing

Remote sensing has many applications in different fields, such as:

- **Agriculture and Soil Science:** we use remote sensing to map soil types, check crop and vegetation health, and assign agricultural productivity.
- **Hydrology and Oceanography:**Remote sensing helps us to estimate water resources, to predict snowmelt-driven runoff, to measure sea surface temperatures, and to track ocean currents.
- **Meteorology and Glaciology:** Remote sensing is used to check the atmospheric temperature and the amount of water vapor in it, to measure the wind speed and direction, and to monitor the movement of the ice in the sea and the glaciers.
- **Geology:** We use remote sensing to identify what type of rocks are in an area, to locate the faults in the earth to map where the minerals are, and to monitor land deformation.
- **Disaster Management and Environmental Monitoring:**emote sensing helps us to warn people early if there is going to be a flood, storm, sandstorm or wildfire. It also helps us to monitor environment pollutions and track land degradation.
- **Defense and Military:** Remote sensing is used to help with surveillance and reconnaissance to map the land and important locations, to monitor the borders and to detect if there are any changes in the buildings or if the troops are moving.

2.7 Electromagnetic Radiation (EMR) and the Spectrum

Remote sensing works by detecting and recording electromagnetic radiation (EMR) that has been either reflected or emitted from the Earth's surface. This energy moves through space at the speed of light, which is approximately $c \approx 3 \times 10^8$ m/s. [33]

2.7.1 Wave and Particle Duality

To understand sensing we need to look at electromagnetic radiation (EMR) in two different ways:

- **Wave Theory:** This theory says that electromagnetic radiation (EM) as energy that travels in a harmonic sinusoidal pattern, and the relationship between velocity (c), frequency (ν), and wavelength (λ) is defined by the following expression:

$$c = \nu\lambda \tag{2.1}$$

[34]

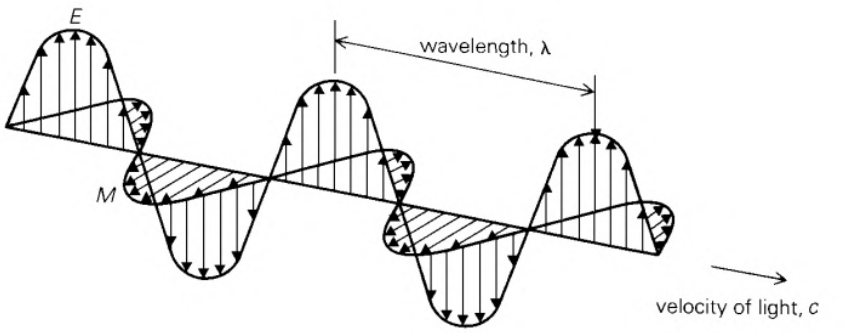


Figure 2.5: Electromagnetic wave propagation showing wavelength (λ) and velocity (c) [34].

- **Particle Theory (The Quantum Model)** This theory says that electromagnetic radiation (EMR) is made up of units of energy called photons or quanta. According to this theory:
  - **Energy Units:** We measure Electromagnetic energy in units like Joules (J) or electron volts (eV) [35].
  - **Radiant Flux:**The rate at which the energy moves from one place (such as the Sun) to another (such as the Earth), and it is defined as flux, measured in Watts (W) [35].
  - **Photon Properties:** Photons are massless particles, and they move at the speed of light, approximately $299,792.46$ km/s [35].

  The energy of a single photon (Q) is defined by a simple relationship called the quantum relationship:

$$Q = hf \tag{2.2}$$

  By substituting the relationship $\lambda = \frac{c}{f}$, the wavelength associated with a quantum of energy is defined as :

$$\lambda = \frac{hc}{Q} \quad \text{or} \quad Q = \frac{hc}{\lambda} \tag{2.3}$$

  Where h is Planck's constant (6.626×10^{-34} J·s) and f is the frequency [35]. This relationship confirms that the energy of a proton is inversely proportional to its wavelength. As a consequence, electromagnetic radiations (EMR) with shorter wavelengths are more energetic than electromagnetic radiations with longer ones. [35].

2.7.2 Principal Divisions of the Electromagnetic Spectrum

The electromagnetic spectrum is the result we get from the decomposition of radiation into frequencies. It includes a range of wavelengths from the small ones like picometers to the big ones like kilometers. When we use sensing we usually divide the spectrum into these main parts:

- **Ultraviolet (UV) Region (0.30 μm - 0.38 μm):** This part of the spectrum is beyond the violet light that human eyes can see; this radiation is scattered by the Earth's atmosphere, and it is generally not used for surface imaging.
- **Visible Spectrum (0.4 μm - 0.7 μm):** This is the part that human eyes can see. It is composed of colors like blue (0.446 - 0.5 μm), green (0.5 - 0.578 μm), and red (0.62 - 0.7 μm); the light that an object reflects determines its color.
- **Infrared (IR) Spectrum (0.7 μm - 100 μm):** This part was discovered by William Herschel in 1800. It includes two types of radiation: reflected infrared (0.7 - 3.0 μm) and thermal infrared (3 - 35 μm) which is the heat that objects give off.
- **Microwave Region (1 mm - 1 m):** We use these wavelengths in sensing. They can pass through clouds, which makes them really useful.
- **Radio Waves ($>$ 1m):** This is the part of the electromagnetic spectrum. We mostly use it for commercial broadcast and meteorology.

2.7.3 Radiation-Matter Interactions

When electromagnetic radiation (EMR) hits a surface or medium, the energy (E_i) it carries must be conserved. The Law of conservation of energy says that the energy that hits the earth interacts in three different ways: Reflection (E_r), Absorption (E_a), and transmission (E_t) [36]. The principle of energy conservation says that the sum of these parts must equal one. This is shown in the equation below:

$$E_i(\lambda) = E_r(\lambda) + E_a(\lambda) + E_t(\lambda)$$

Where:

$$E_I(\lambda) = \text{incident energy}, \tag{2.4}$$
$$E_R(\lambda) = \text{reflected energy}, \tag{2.5}$$
$$E_A(\lambda) = \text{absorbed energy}, \tag{2.6}$$
$$E_T(\lambda) = \text{transmitted energy}. \tag{2.7}$$

[37]

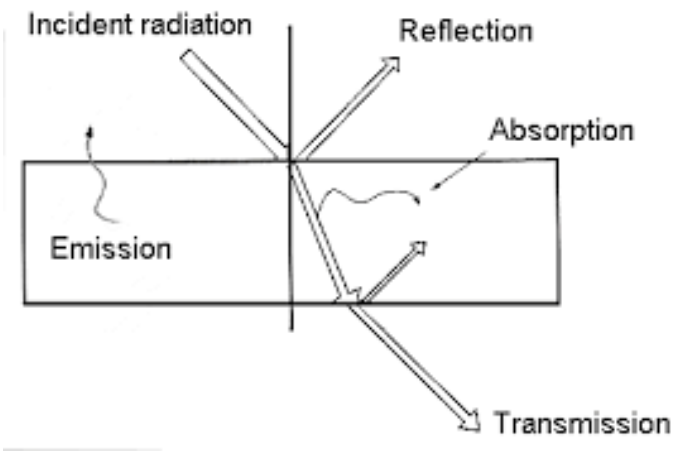


Figure 2.6: Reflection, Absorption and Transmission

Source: https://share.google/EwRp1BdKKzXyZghg6

In the context of remote sensing, we think of these interactions as simple numbers that show us what fraction of energy is involved compared to the incident radiation. The principle of energy conservation says that the sum of these components must equal one, as shown in the equation below:

$$\alpha(\lambda) + \tau(\lambda) + \rho(\lambda) = 1 \tag{2.8}$$

Where each term means the following: Where:

- $\alpha(\lambda)$ denotes the **Absorptance**.
- $\tau(\lambda)$ denotes the **Transmittance**.
- $\rho(\lambda)$ denotes the **Reflectance**.

1. **Absorption:** Absorption (A) happens when energy, from radiation, gets into the material. [38].
2. **Transmission:** Transmission (T) happens when electromagnetic radiation goes through the atmosphere or some materials and is not absorbed or scattered. [39]
3. **Reflection:**Reflection (R) happens when radiation hits a surface; it changes direction[40].

2.8 Conclusion

Remote sensing is an important technology for observing the Earth without direct contact.

In this chapter we have introduced the fundamental ideas of remote sensing, starting with its definition, evolution, workflow, components, and applications, and we have also talked about the role of electromagnetic radiation in data acquisition.

Because sensors and computers are getting better, remote sensing can now give us a lot of useful information for many different areas. One of the most important thing that remote sensing gives us is satellite imagery, which plays an important role in analysis and interpretation.

The next chapter is about satellite imagery, its characteristics, and its importance in modern image analysis.

Chapter 3

Satellite Imagery

3.1 Introduction

Satellite imagery plays an important role in modern remote sensing; it helps us to observe and understand the Earth's surface and atmosphere. We get a lot of useful information from these images about land cover, oceans, and weather by capturing electromagnetic energy that is reflected or emitted from the Earth at different scales of distance and time.

These images help us to watch areas all the time, which is so hard or impossible to do by looking at the ground alone.

Now the satellite technology and sensor design become much more detailed and diverse; this includes visual, thermal, and microwave data. So, as a result, we need ways to prepare and analyze these large amounts of data.

This chapter presents the basics of satellite imagery. It covers types of images, sensor characteristics, preprocessing steps, and analysis methods and helps us understand their importance in different remote sensing applications.

3.2 Fundamentals of Satellite Imagery

3.2.1 Definition of Satellite Imagery

Satellite imagery gives us spatial and temporal information about the environment that we cannot get from traditional methods. When we look at areas of the Earth, many times we see that the Earth is like a big system where this system is composed of many different parts, such as the atmosphere, oceans, and land surfaces, all working and interacting together. This way of looking at the Earth helps us learn how these components influence each other to keep the environment in balance [41].

Figure 3.1: Satellite Imagery

Source: https://share.google/G6vLYcSXxJMIVllpN

3.2.2 Digital Representation of Satellite Images

Satellite images are digitally represented as a grid of pixels, where each of these pixels shows an area on the Earth. Each pixel contains a value or a number; this value is called a digital number (DN). The digital number tells us how strong the electromagnetic radiation is in an area. Digital representation allows for efficient storage, processing, and analysis of satellite data by using computers and Geographic Information Systems (GIS).

3.2.3 Types of Satellite Images

Satellite images can be classified based on the kind of sensor used and the spectral characteristics of the satellite images:

- **Panchromatic Images:** Panchromatic imagery is about black-and-white images. These pictures are taken over a large range of light; usually this range is in the light spectrum from blue to red. Panchromatic images combine all the visible light into one high-resolution image[42]. Panchromatic images are really good because they are clear and detailed, and they show a lot of things in one image.

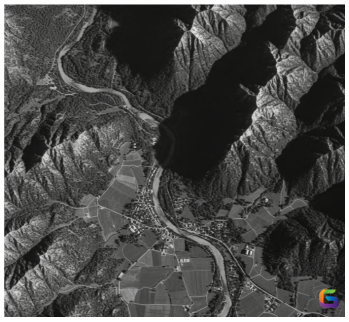


Figure 3.2: Panchromatic Image

[42]

- **Multispectral Images:** A multispectral image is made up of separate image layers or bands of the same place taken at the same time by a sensor across different regions of the electromagnetic spectrum. This includes the visible, near-infrared (NIR), and shortwave infrared (SWIR) ranges. These layers are taken at separate wavelengths, so they give us the information we need for land cover classification and environmental analysis. [43]

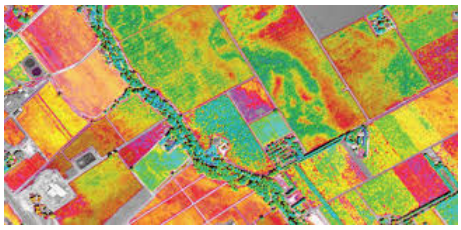


Figure 3.3: Multispectral Image

Source: https://share.google/images/hYr5uVoi8M7lvyILP

- **Hyperspectral Images:** Hyperspectral images are images where a continuous spectrum is measured for each pixel [44], this allows for identification of materials based on their spectral signatures. They are especially useful for analysis of surface composition and the environment. These images help identify materials accurately using their signatures.

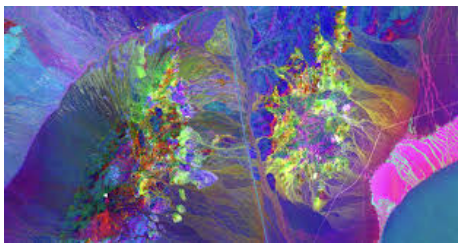


Figure 3.4: Hyperspectral Image

Source: https://share.google/GnkyBmdTrarZETWNU

- **Thermal Images:** Thermal infrared sensors detect and capture the electromagnetic radiation that the Earth's surface emits, allowing the estimation of the temperatures of the surface and helping in analyzing thermodynamic phenomena. Thermal images are taken to see visible and invisible objects that we cannot see when the light is not good[45].

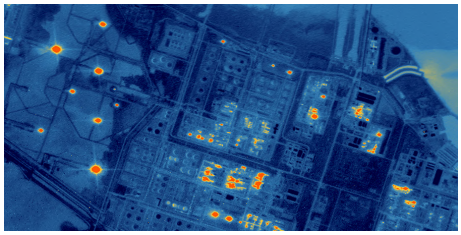


Figure 3.5: Thermal Image

Source: https://share.google/c9zVDehWQY6ujbh7g

- **Radar (SAR) Image:** Synthetic Aperture Radar (SAR) imagery represents a very powerful and capable technology in earth observation and geospatial intelligence.[46]. SAR systems use microwave sensors that send and receive their own signals; this allows them to work on their own, not relying on sunlight or weather conditions, and as a result they can provide images in any weather, day or night.

Figure 3.6: Radar (SAR) Image

Source: https://share.google/jrYa9LjrADvvrENYk

- **Optical Image:** Optical images capture light in the visible colors, also in the infrared part of the spectrum. These images can be shown as either natural or false-color images. Natural color images are what we can see with our eyes, and false-color images use infrared information to highlight different features. This makes them very useful for analyzing agricultural lands, vegetation health, and soil composition. [47].

Figure 3.7: OPtical Image

Source: https://www.viridiengroup.com/expertise/satellite-mapping/optical-satellite-imagery

3.2.4 Characteristics of Satellite Images

The main characteristics that define satellite images are:

- **Spatial Resolution:** Spatial resolution is about how detailed a satellite image can be. It is the size of a pixel on the ground; a high spatial resolution means each pixel represents an area, like 1 meter by 1 meter, which gives a detailed pictures of the earth's surface. On the other hand, a low spatial resolution means each pixel is bigger, like 30 meters by 30 meters, in which gives a general view; in this case, small details can be hard to see. [48]

Table 3.1: Classification of Satellite Image Spatial Resolution.

Resolution Level	Spatial Resolution	Example Satellites
Low Resolution	>30 m/pixel	Landsat 1–5, MODIS (Terra/Aqua)
Medium Resolution	5–30 m/pixel	Sentinel-2, Landsat 8
High Resolution	1–5 m/pixel	IKONOS, RapidEye
Very High Resolution	<1 m/pixel	WorldView (Maxar), up to 0.46 m

- **Spectral Resolution:** Spectral resolution is about a sensor's capacity to address small differences in light wavelengths by using narrow spectral bands. while standard multispectral instruments use 3 to 10 bands. Hyperspectral systems use hundreds or thousands of narrow bands and give detailed spectral signatures. For instance, the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) is an example; it measures how much light is reflected in 224 bands that are right next to each other. This level of detail helps us see things on the surface; these things include types of rocks, what minerals they are made of, types of vegetation, and other features. In practice, high spectral detail can spot small changes in biology or chemistry. These changes can be things like a lot of pigment in one place or small living things in a type of light. [49]

- **Radiometric Resolution:** Radiometric resolution defines how good a sensor is at detecting changes in energy; it depends on how many bits of data are assigned to each pixel. This resolution is shown as a power of two, like 2^n. For instance, a sensor that uses 8 bits can show 256 levels (from 0 to 255) for each pixel signal. A sensor with a higher radiometric resolution allows for detecting differences in how much light or energy is reflected or emitted by surfaces. This is really important for applications such as water quality assessment, where it helps differentiate minor variations in ocean color and how much stuff is suspended in the water. [49]

- **Temporal Resolution:** Temporal resolution is the time that the earth takes for an observation satellite to come back to the same location. This is like one circle that the satellite makes around the earth. Temporal resolution can be very different for satellites. The Landsat-8 satellite takes 16 days. The Sentinel-2 satellites take 10 days. The MODIS satellite can do it every day. [50].

Table 3.2: Summary of Satellite Imagery Resolutions.

Resolution Type	Measurement Unit
Spatial	Meters (m)
Spectral	Wavelength (μm)
Radiometric	Bits (e.g., 8-bit, 12-bit)
Temporal	Days / Hours

3.2.5 Digital Image Representation Formats

The Earth observation work starts with the conversion of the information from sensors into data. Remote sensing instruments send digital imagery to the ground stations all the time; these images are then processed and displayed. The way we process these pictures has been getting better for over thirty years. [51].

Today digital images usually come in three formats; these formats are based on how they represent pixel values and color:

- **Binary Images (Black and White):** Binary image formats have only two colors, which are black and white; each little part of the binary image formats, called a pixel, is made up of one bit. They are often generated using a process called "thresholding" and are widely used in a lot of applications such as document scanning, barcode recognition, and Optical Character Recognition (OCR)[52].

Figure 3.8: Binary Images

Source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQbUBG_Ivv8yZlvxndJwVig2oxh1UVWCPA8lQ&s

- **Grayscale Images (Niveau de Gris):** Grayscale images have shades of gray from pure black to pure white. In digital images, each pixel in grayscale shows how bright or dark it is; this brightness is usually measured as a number from 0 to 255, where 0 is black and 255 is white. This format is widely used to reduce complexity and simplify image data while reducing the amount of data details, making it highly useful in many applications such as document processing, digital photography, and medical diagnostics. [53].

Figure 3.9: Grayscale Images

Source: https://cdn.serc.carleton.edu/images/research_education/geopad/san_ysidro_nm.jpg

- **RGB Composite Images:** A true color image is generated by combining the red, green, and blue (RGB) spectral bands. This format of three channels is what deep learning models use, as it captures the necessary spectral and textural information required to classify the land-cover in the LoveDA dataset. The LoveDA dataset uses this format of green, red, and blue colors to make sure the models can classify land-cover classses[54].

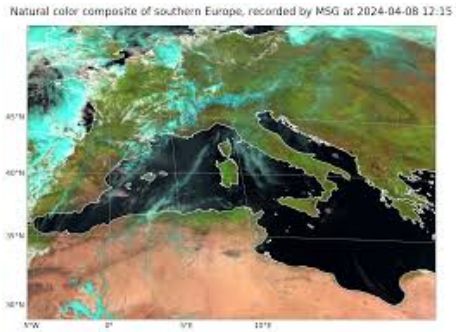


Figure 3.10: RGB Composite Image

Source: https://dust.trainhub.eumetsat.int/_images/0b51b7278c99e1d1f4fa1d1be0069f8b2194af42e9b9fa92e4f919c92bbfac91.png

3.3 Preprocessing of Satellite Images

3.3.1 Radiometric Correction

Radiometric correction was applied to fix the differences in pixel values that happened because of problems with the sensor and the air. This preprocessing step was used to convert the digital numbers (DN) of satellite images into physical units that measure the radiation that the sensor detects, which are then converted to surface reflectance [55].

3.3.2 Geometric Correction

Geometric correction is an important step that we need to do before we can use the data from satellites and other sensors. This step helps to eliminate distortions in remotely sensed data; this means that the pixel should have the coordinates, which are the *xandy* values. Geometric correction is necessary so that we can combine the sensing data with other types of data in a geographic information system (GIS) environment [56].

3.3.3 Atmospheric Correction

The Earth's surface has its special way of reflecting light. When this light travels through the air, it gets changed. This is because the air scatters and absorbs the light. So we do what's called "atmospheric correction"; this helps us to eliminate the effects that the air has on the light. The goal of this correction is to find out the true surface reflectance; this reflectance tells us about the properties of the earth's surface. [57],ensuring that the data we are looking at represents the ground objects rather than just atmospheric interference.

3.3.4 Image Enhancement

Image enhancement is a way to make pictures look better by changing how bright or dark each part of the image is. This process is about varying the pixel intensity of the input images[58]to highlight specific features, this enhancement of digital values improves the visual interpretability of the imagery, facilitating the identification and analysis of surface features(soil, water, and vegetation...)

3.4 Image Segmentation

3.4.1 Definition

Image segmentation is an essential task in computer vision. It aims to partition an image into multiple meaningful regions, each pixel of which is labeled to indicate the object or region it belongs to. Image segmentation can be treated as a classification problem of pixels with semantic labels or as a partitioning of individual objects as described by Minaee et al. (2021) [59]. This process plays a central role in a broad range of applications, including medical image analysis, autonomous vehicles, video surveillance, and remote sensing. The goal is not simply to detect what is in an image, but to precisely locate where each object or region is at the pixel level, which makes it a significantly more challenging task than image classification.

3.4.2 Types of Image Segmentation

There are three different algorithms for the image segmentation algorithm.
as illustrated in Figure 3.11:

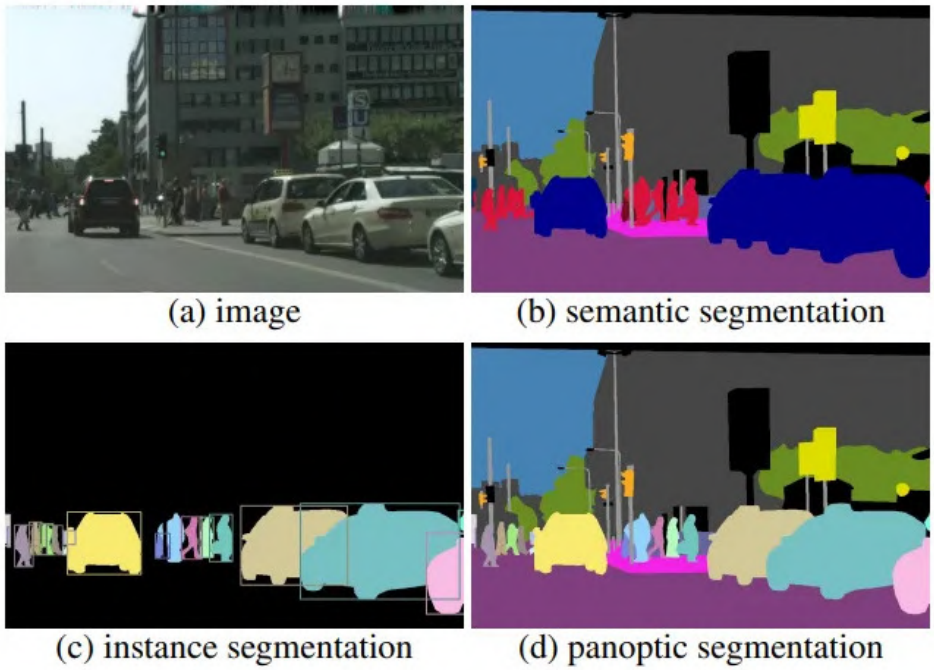


Figure 3.11: The three main types of image segmentation.
Source: https://medium.com/ching-i/image-segmentation-semantic-segmentation-1-eb727d2b132f

1. **Semantic Segmentation:**Semantic segmentation involves associating each pixel of an image with a specific class label, ensuring that all pixels belonging to the same category are grouped together. However, it does not distinguish between individual instances; for example, two separate buildings would be assigned the same label, treating them as a single semantic entity rather than distinct objects [60]

2. **Instance Segmentation:** When we look at semantic segmentation, it sees all objects of the same class as the same. Instance segmentation is different. It's detecting every individual object separately. Instance segmentation combines the tasks of object detection and semantic segmentation to assign a unique mask to each distinct instance (e.g., telling apart "car A" and "car B" rather than labeling them as a single class) [61].

3. **Panoptic Segmentation:** Panoptic segmentation is the unified version of both semantic and instance segmentation. It assigns a class label to every pixel in the image while simultaneously distinguishing between individual instances of countable objects. As defined by Kirillov et al. (2019), it provides a complete and coherent scene understanding by combining the advantages of both previous approaches [62].

3.4.3 Semantic Segmentation of Satellite Imagery

Land cover semantic segmentation in remote sensing is about figuring out what type of land cover is at every image pixel. As the spatial resolution of satellite images gets

higher, more urban objects are now clearly visible in satellite images, and studies have shifted their paradigm from spectral image classification and pixel-based image analysis to pixel-level semantic segmentation [63].However, this task presents specific challenges in real-world remote sensing scenes:

A. Multi-scale Object Variability : Objects in the same category appear in completely different geographical landscapes in different scenes, which increases the scale variation. For example, buildings in urban areas have large size variance, while buildings in rural areas are mostly small scale. If models are to handle multi-scale objects, they need to be able to capture at multiple scales [63].

B. Complex Background Samples : The remote sensing semantic segmentation task is always faced with complex background samples, that is, land-cover objects that are not of interest. In other words, these are ground elements that we are not trying to find., such as undefined surfaces, but which still occupy large portions of the image and can be easily confused with meaningful classes[63].

C. Class Imbalance Problem : In satellite imagery, some land-cover classes occupy significantly more pixels than others, making it difficult for models to learn minority classes effectively. This problem is particularly present in urban remote sensing images where small objects such as narrow roads are easily dominated by large homogeneous regions such as vegetation or background [64].

3.5 Classical Approaches to Image Segmentation

Before deep learning took over people used methods to segment satellite images. These methods came from image processing.They are worth looking at briefly because they had some limitations.These limitations are why people moved to deep learning solutions, for satellite image segmentation now.

3.5.1 Edge-Based Segmentation

Edge-based segmentation works by locating boundaries between regions, looking for abrupt changes in pixel intensity.

- **Edge Detection Operators:** Operators such as Sobel and Prewitt estimate image gradients to capture intensity variations. The Canny algorithm builds on these by chaining together Gaussian smoothing, gradient computation, non-maximum suppression, and hysteresis thresholding, yielding edges that are both thin and well-connected [65].
- **Active Contours:** Introduced by Kass et al., Snakes deform a curve by minimizing an energy function that trades off smoothness against attraction to image gradients. Level Sets extend this idea by representing the curve implicitly, which allows it to handle topological events such as splitting or merging without special treatment, making them more suitable for complex structures [66].

3.5.2 Region-Based Segmentation

Rather than seeking boundaries, region-based methods group pixels together according to how similar they are.

- **Region Growing:**Beginning from a seed pixel, the method expands outward by absorbing neighbors that share similar properties such as intensity or texture. It performs well on homogeneous regions, though it remains sensitive to noise and to the initial choice of seed [67].
- **Split and Merge:** The image is first recursively subdivided into homogeneous blocks, which are then merged whenever adjacent blocks prove sufficiently similar. Although the approach captures spatial structure reasonably well, it is sensitive to the similarity criteria chosen, and poorly tuned parameters can easily lead to over ,or under-segmentation [68].

3.5.3 Thresholding Methods

Thresholding is the most straightforward segmentation strategy, assigning each pixel to a class based solely on its intensity.

- **Otsu Thresholding:** Proposed by Otsu in 1979, this method automatically determines an optimal global threshold by maximizing the variance between two pixel classes. It is non-parametric and requires no manual intervention, but performs poorly on images with uneven illumination or low contrast [69].
- **Adaptive Thresholding:** Instead of applying a single threshold globally,this approach derives a local threshold for each pixel from its surrounding neighborhood, giving it greater resilience to spatial illumination changes [70].

These old ways of doing things have a problem: they rely on rules that people made and simple ideas about what pixels look like. The thing is, these rules and ideas do not work well when we look at the different kinds of pictures that satellites take nowadays. Satellite pictures are very detailed and have a lot of things going on. The old rules just do not cut it. This is why people started using deep learning, which is what the next part is, about.

3.6 Conclusion

Satellite imagery is the core of remote sensing; it gives us a detailed view of the surface of the earth. In this chapter, we presented the different types of satellite imagery, their characteristics, and the essential preprocessing steps needed to ensure we get the right information from them. We also introduced semantic segmentation as the analytical task that assigns a meaningful label to every pixel of a satellite image, and highlighted the main challenges it faces in remote sensing scenes, such as multi-scale object variability, complex backgrounds, and class imbalance.

Addressing these challenges requires powerful and intelligent methods that go beyond traditional image processing. The next chapter presents the deep learning

techniques that we use in this work, and explain how Convolutional Neural Networks and Vision Transformers can be combined to give a hybrid architecture, to perform semantic segmentation accurately and efficiently on high-resolution satellite imagery.

Chapter 4

Deep Learning

4.1 Introduction

The automatic analysis of high resolution satellite imagery is a revolutionary step forward in remote sensing, allowing the precise detection of land cover patterns and environmental changes on a scale not possible through traditional visual interpretation. Manual analysis of such imagery is difficult and time consuming, and is subject to the personal interpretation of the analyst and limited in capturing complex spatial relationships in urban, agricultural and natural landscapes. As a result, advanced computational techniques have been developed to ease the burden on analysts and to improve the reliability of large scale geographic monitoring.

With the combination of machine learning and artificial intelligence, modern deep learning architectures enable computers to automatically extract meaningful features and semantic information from raw satellite data. These advances are critical. The use of convolutional neural networks for local feature extraction and Transformer-based models for global contextual reasoning allows automated systems to reach unprecedented accuracy in pixel-wise land-cover classification. This accelerates the mapping of critical infrastructure, vegetation and water bodies, and reduces the human error in applications from urban planning to disaster response.

This chapter establishes the theoretical foundations that support our hybrid approach. We begin by outlining the hierarchy of Artificial Intelligence, Machine Learning, and Deep Learning, then examine the architectural principles of CNNs and Vision Transformers, their strengths, limitations, and the motivation for their fusion. Key strategies such as transfer learning and regularization are also discussed, along with the standard evaluation metrics used to measure segmentation performance.

4.2 Artificial Intelligence

4.2.1 Definition

Artificial Intelligence (AI)is the field of science and engineering that deals with making machines that can think like people.

These tasks include thinking and learning and seeing things and solving problems and understanding language and making decisions .

John McCarthy coined the term in 1955 and proposed a foundational definition:

> "Artificial Intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs."

Machine Learning (ML) and Deep Learning (DL) are two important subfields As shown in Figure 4.1 [71].

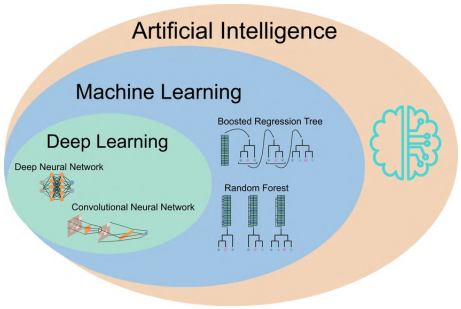


Figure 4.1: Relationship between AI, ML, and DL [72].

Artificial intelligence begins with the construction of expert systems shown in Figure 4.2 .

In the 1970s and 1980s expert systems were the starting point for Artificial Intelligence. Expert systems are complicated because they try to make decisions like an expert in a specific area [73].

Expert systems use rules that people write by hand. These rules are very clear. They work well for problems that are easy to understand like figuring out what is wrong with someone when they are sick for example with a system called MYCIN 4.3[74],but they have some problems:

Knowledge acquisition bottleneck: It takes a lot of work to encode expert knowledge, and it's hard to make it work for a lot of people.

Brittleness: when expert systems face a situation that is not covered by the rules they do not work well.

Complexity of maintenance: It is hard to update and check large rule bases [73] .

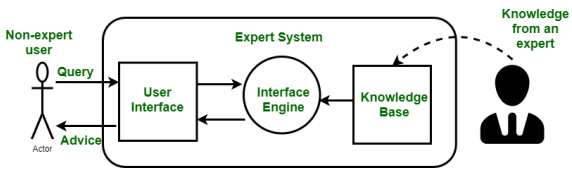


Figure 4.2: expert system [75].

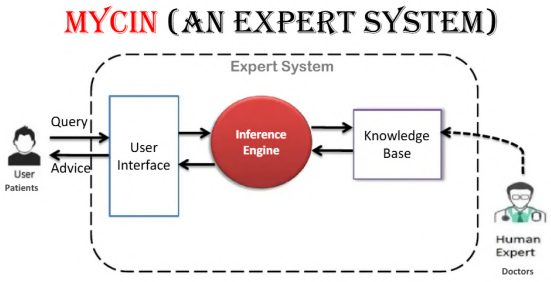


Figure 4.3: MYCIN expert system for medical diagnosis. [76].

4.3 Machine Learning (ML)

4.3.1 Definition

A branch of artificial intelligence called machine learning (ML) allows systems to recognize patterns in data without needing to be specifically programmed for every task. An ML model improves its predictions as it is exposed to more data by modifying its internal parameters based on training examples instead of depending on hand-coded rules. It incorporates techniques such as: supervised learning, unsupervised learning, and reinforcement learning (Figure 4.4)[77].

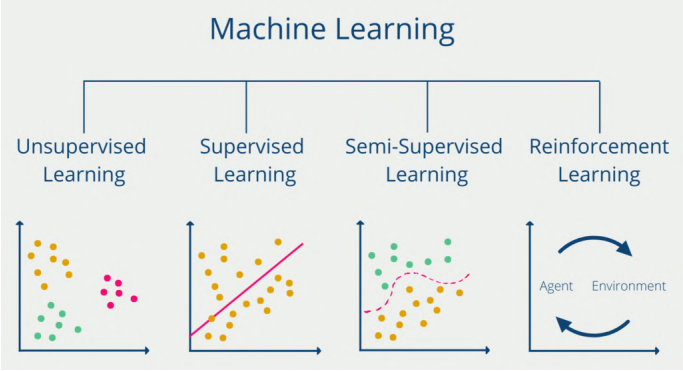


Figure 4.4: Different types of machine learnings [77].

Let us detail each of these techniques as follows:

- **supervised learning** is The most common way to learn;through The model learns from a labeled dataset, which means that each input is linked to a known output label. There are two main types of tasks: classification, which predicts a specific category (like spam detection or digit recognition), and regression, which predicts a continuous value (like estimating the price of a house). This project treats semantic segmentation as a supervised pixel-wise classification task. Each training image is paired with a fully annotated ground truth mask from the dataset [77].
- **Unsupervised learning** trains the model using data that doesn't have any labeled answers. The system finds hidden structure directly from the input in

two main ways: Clustering is when you put samples that are similar together (like k-means). Association learning finds connections between variables in big datasets. Unsupervised methods are good for exploratory analysis of satellite data, but they don't work well for precise land-cover segmentation, which needs pixel-level supervision [77].

- **Semi-Supervised Learning** represents a hybrid paradigm that bridges supervised and unsupervised learning, It uses both labeled and unlabeled data to train, usually by mixing a small number of labeled examples with a much larger number of unlabeled examples. The model generalizes better when it uses unlabeled data because it takes advantage of the underlying structure and distribution of that data. It does this by making assumptions about things like smoothness, clustering, and bifurcation properties. This method greatly increases the accuracy of learning while making labeling less of a hassle. This is important in fields where labeling is expensive or takes a lot of time [78].
- **Reinforcement Learning** In RL, an agent learns how to make the best choices by interacting with its environment using rewards and punishments. The agent tries out different strategies to see which ones will give them the most long-term rewards. It works especially well for tasks that need to be done in order, like robotics and self-driving cars [79].

4.4 Deep Learning

In traditional machine learning, feature extraction and classification are performed separately. Before training a classifier, domain experts need to manually design and extract useful features from raw data. For example, they need to make texture descriptors for images or spectral indices for satellite images. Then, the model learns how to make predictions by using these handcrafted features. But this process of feature engineering takes a lot of work, time, and expert knowledge, especially when working with large, high-dimensional data like very high-resolution satellite images. Deep learning fixes this problem by combining feature extraction and classification into one trainable framework that works from start to finish (Figure 4.5) [80].

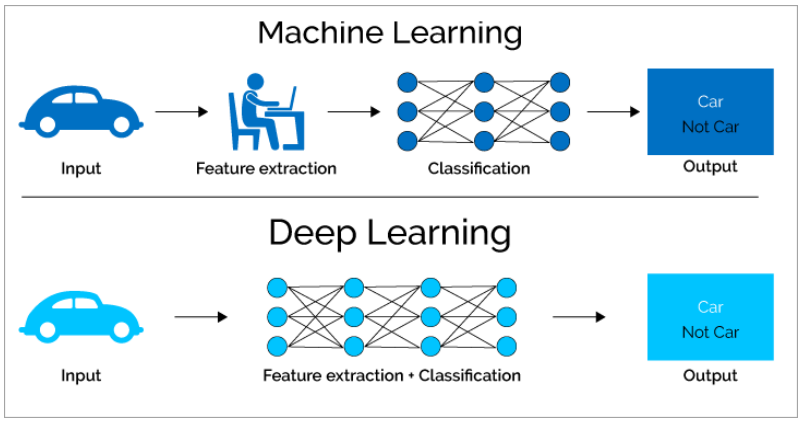


Figure 4.5: Comparison between traditional machine learning and deep learning [81].

Artificial Neural Networks (ANNs) are at the heart of deep learning. Loosely inspired by the brain's network of neurons, ANNs are computational systems[82] . These models are composed of layers of simple units, or neurons, each of which computes a weighted sum of its inputs and applies a non-linear activation. If you stack a lot of these layers, the network learns to represent the data on increasingly abstract levels. The first layers will tend to represent simple things like edges or color contrasts. The deeper layers will combine these to represent more complex things, like textures, parts of objects, and eventually whole objects or scenes. It is this hierarchical learning that allows deep networks to directly map raw inputs to semantic outputs without manual feature design[83]. However, fully connected ANNs treat each input dimension independently, ignoring the spatial relationships fundamental to images. This leads to an explosion of parameters and poor generalization on tasks that are visual. To solve these limitations, architectures that included Fully Convolutional Networks (FCN) and Convolutional Neural Networks (CNN) were implemented. CNNs are able to learn spatially aware features efficiently by enforcing local connectivity, sharing weights across spatial locations, and preserving the grid structure of images. We will explore this in detail next.

4.4.1 Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are a popular type of ML model that have been widely utilized for various tasks, including image recognition, speech recognition, and NLP[84].

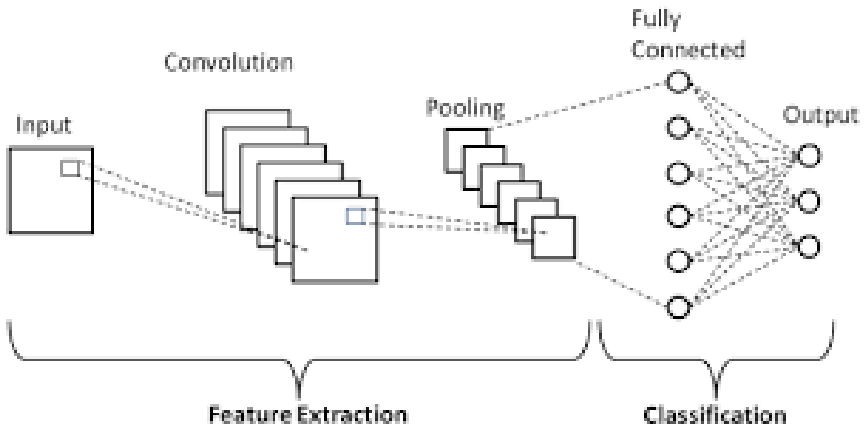


Figure 4.6: Convolutional Neural Networks (CNNs)
Source: https://www.researchgate.net/figure/Basic-structure-of-Convolutional-Neural-Network-CNN-23-Artificial-neural-network_fig1_353049916

In recent years, convolutional neural network (CNN)-based deep learning algorithms have achieved a series of breakthrough research results in the fields of object detection, image semantic segmentation, and image classification[85]. Their efficiency is derived from three fundamental operations:

4.4.1.1 Convolutional Layer

The convolution operation is one of the fundamental building blocks of a convolutional neural network[86].

The convolution layer (CONV) uses filters that perform convolution operations as it is scanning the input I with respect to its dimensions. Its hyperparameters include the filter size F and stride S. The resulting output O is called a feature map or activation map. The convolution step can be generalized to the 1D and 3D cases as well. [87]

At the start of the convolution, the filter window is positioned in the top-left corner of the image. It then moves a number of pixels (the stride) to the right. When it gets to the end of the row, it moves down by one stride and keeps scanning left to right. This sliding continues until the filter has traversed the entire input image, producing a full feature map (Figure 4.7)[82].

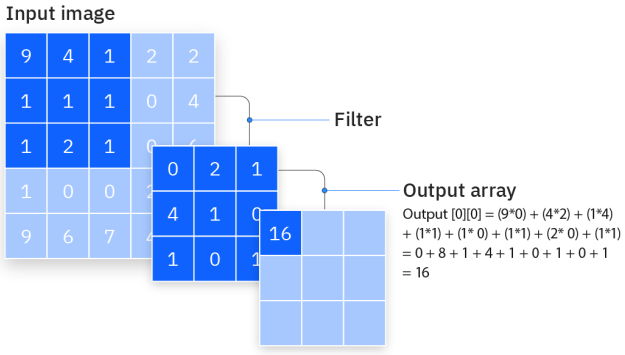


Figure 4.7: Convolutional Layer

Source: https://www.ibm.com/think/topics/convolutional-neural-networks

4.4.1.2 Pooling Layers

The second important step in building convolutional neural networks is pooling, which is usually inserted between two convolutional layers to perform spatial downsampling. Pooling is a method of feature map dimensionality reduction that collapses multiple activations into fewer values, instead of processing every single value. It keeps the most important information. This downsampling is useful for two main reasons . Firstly it reduces the computational cost of the network by reducing the number of parameters . Secondly it allows for a form of spatial invariance which can help control overfitting , in that the network is still able to recognize features even if they move slightly in position . There are different pooling techniques like Max Pooling, Average Pooling, and Min Pooling. But Max Pooling is the most prevalent choice in practice. It works by selecting only a maximum activation value of each local window and discarding the others, which is surprising effective at preserving the most important features (such as strong edges or distinctive textures) while filtering out less relevant details (Figure 4.8)[88].

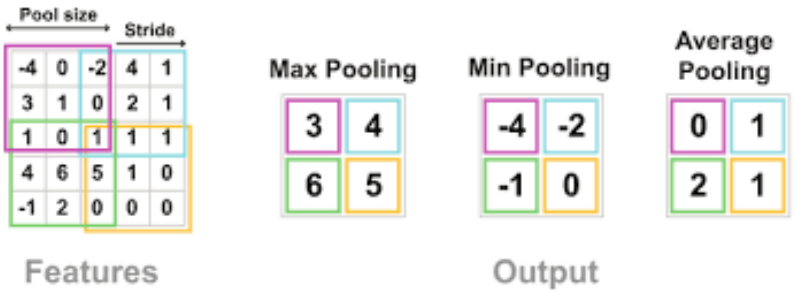


Figure 4.8: Pooling Layer

Source: https://epynn.net/Pooling.html

4.4.1.3 Fully Connected (FC)

The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons. If present, FC layers are usually found towards the end

of CNN architectures and can be used to optimize objectives such as class scores (Figure 4.9) [87].

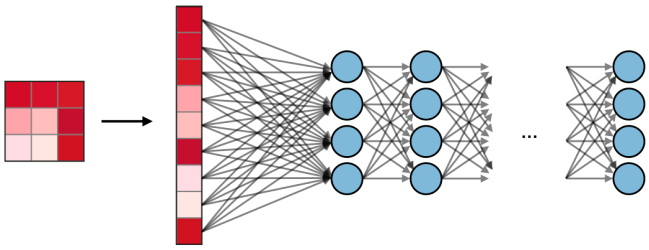


Figure 4.9: Fully Connected (FC)
Source: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks/

4.4.1.4 Activation Functions

is applied to the weighted sum of inputs (before producing the final output of a neuron). It introduces non-linearity and allows the model to learn and represent complex patterns in the data. In the absence of it, even a deep neural network would behave as a simple linear regression model. Activation functions decide whether a neuron gets activated or not based on the weighted sum of inputs and a bias term. They also allow backpropagation. Backpropagation provides gradients for weight updates.

We have many activation functions like ReLU, sigmoid, tanh, softmax, etc (Figure 4.10). Among these, ReLU is the most widely used function in practice [89]. The ReLU function can be represented mathematically using the max() function:

$$A(x) = \max(0, x) \tag{4.1}$$

This means: If x is positive or zero, then $A(x) = x$, otherwise $A(x) = 0$.

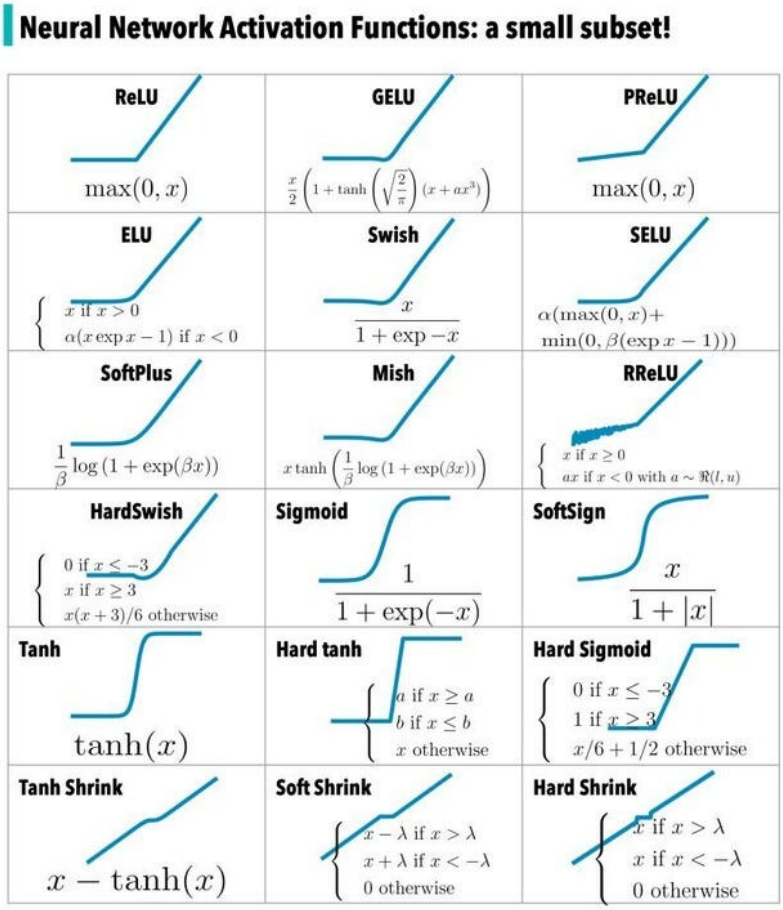


Figure 4.10: Activation functions

Source: https://medium.com/@suthetmilen/why-do-we-use-activation-functions-cant-we-just-stack-layers-cacc5a53f5ee

4.4.1.5 Evolution of CNN Architectures

Currently, the most commonly used algorithms among the innovative techniques of artificial intelligence (AI) are convolutional neural networks (CNNs). Over the decades, different attempts have been made to improve the performance of CNNs, and many well-known architectures have emerged. CNNs are found in many types of architectures, suitable for different tasks such as classification, segmentation, and even object detection as explained in Figure 4.11.

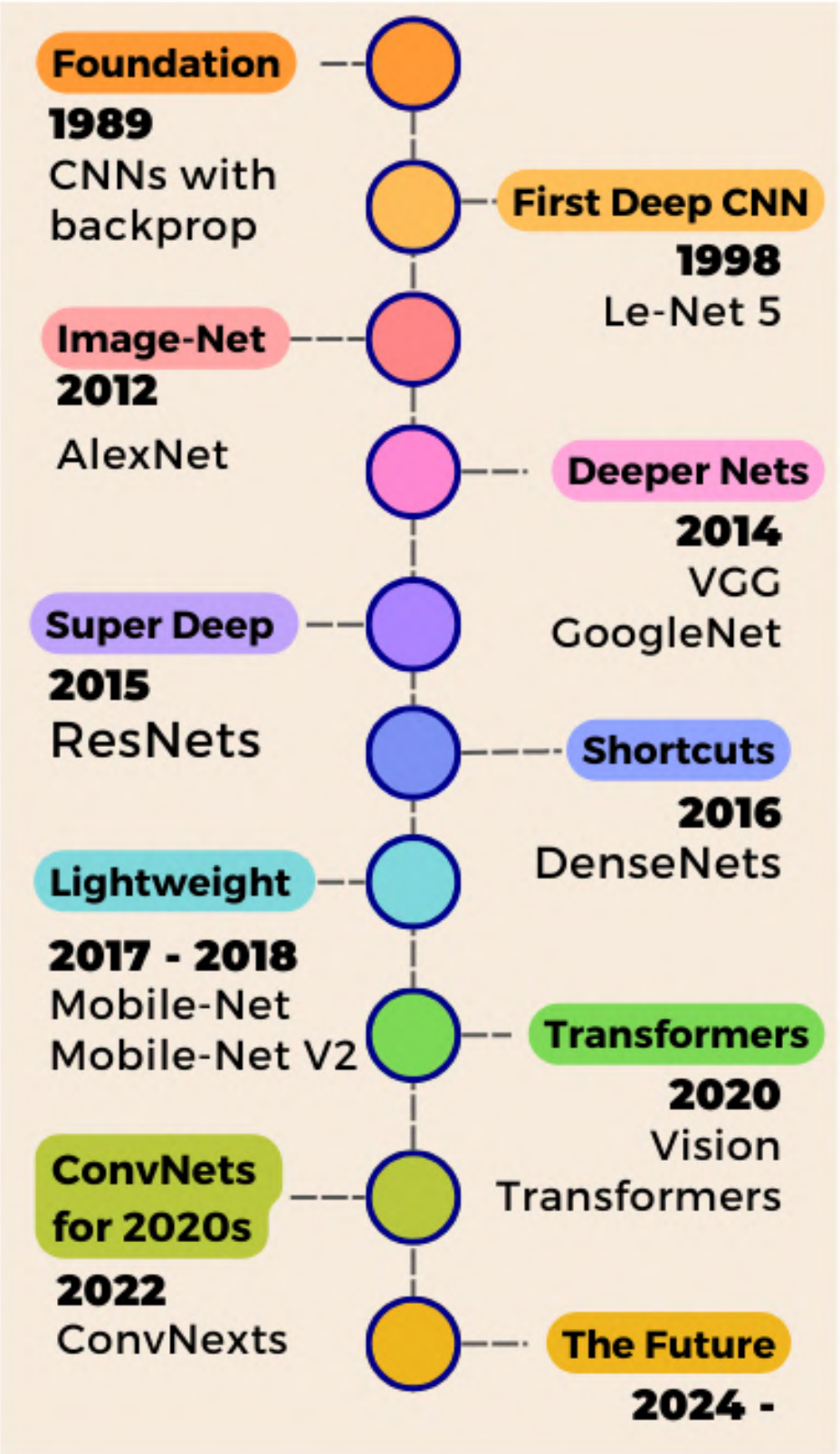


Figure 4.11: Timeline of CNN architecture evolution from 1989 to the present [3]

Source: https://towardsdatascience.com/the-history-of-convolutional-neural-networks-for-image-classification-1989-today-5ea8a5c5fe20/

1. ConvNet (1989) The first convolutional neural network was introduced by LeCun et al. in 1989 at AT&T Bell Laboratories. This pioneering work applied back-propagation to recognize handwritten digits, establishing the fundamental principles of local connectivity and weight sharing that define all modern CNN architecture [90].

2. LeNet (1998) LeNet-5, also proposed by LeCun et al (Figure 4.12) . in 1998, which became the first complete and practical CNN architecture.and introduced the now-standard pipeline of alternating convolutional and pooling layers followed by fully connected layers. LeNet was successfully applied to handwritten digit recognition on the MNIST dataset and is considered the direct ancestor of all modern deep learning architectures [91].

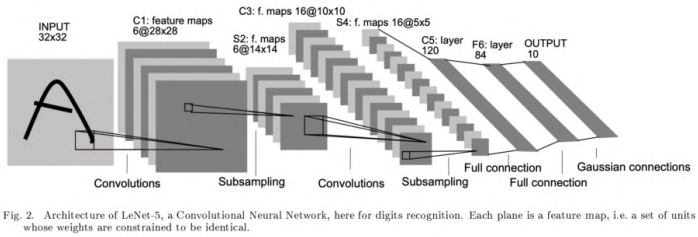


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Figure 4.12: LeNet-5 architecture

Source: https://towardsdatascience.com/the-history-of-convolutional-neural-networks-for-image-classification-1989-today-5ea8a5c5fe20/

3. AlexNet (2012) AlexNet, introduced by Krizhevsky et al. in 2012, marked a major turning point in the history of deep learning. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a significant margin, reducing the top-5 error rate from 26% to 15%. AlexNet was the first CNN to use ReLU activation functions instead of tanh, which greatly accelerated training. It also introduced dropout as a regularization technique to reduce overfitting, and it was the first architecture to be trained on GPUs, making it possible to train much deeper networks than before (Figure 4.13) [92].

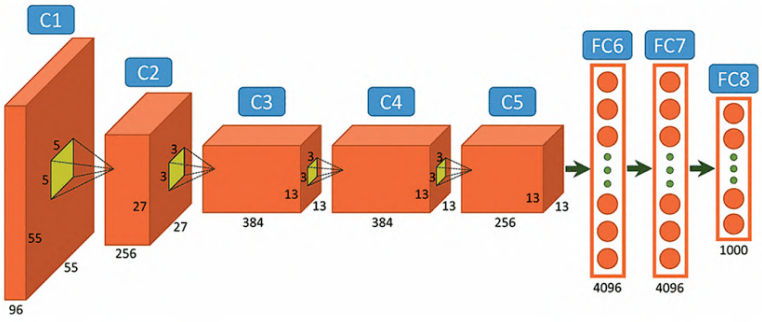


Figure 4.13: The AlexNet Architecture

Source: https://viso.ai/deep-learning/alexnet/

4. VGG (2014) VGG, proposed by Simonyan and Zisserman in 2014, showed that the depth of a network is a critical factor for achieving strong performance. The key idea was to use very small 3×3 convolutional filters stacked deeply, which proved to be much more effective than using larger filters. Despite its simplicity, VGG achieved excellent results on the ImageNet challenge and its architecture became one of the most widely used backbones for feature extraction in many computer vision tasks (Figure 4.14) [93].

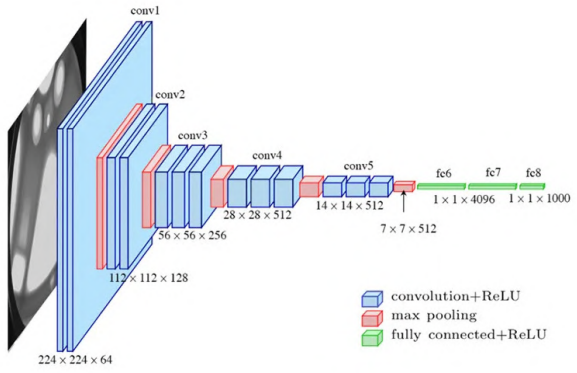


Figure 4.14: The architecture of VGG16

Source: https://medium.com/analytics-vidhya/vggnet-convolutional-network-for-classification-and-detection-3543aaf61699

5. U-Net (2015) While the previous architectures focused on image-level classification, U-Net marked the beginning of a new era dedicated to pixel-level prediction. Introduced by Ronneberger et al. (2015), U-Net was the first architecture specifically designed for semantic segmentation. Built on a symmetric encoder-decoder structure with skip connections, it became the de-facto standard for segmentation tasks and the direct foundation of all modern segmentation architectures (Figure 4.15) [94] [61].

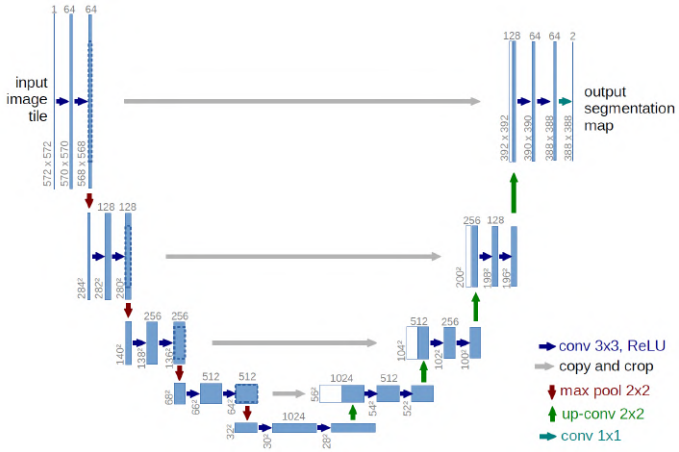


Figure 4.15: U-Net: a revolutionary architecture for image segmentation

Source: [94]

6. ResNet (2015) While U-Net established the segmentation framework, deep networks still faced a fundamental training problem: as networks grew deeper, gradients would vanish during backpropagation, preventing effective learning. ResNet, introduced by He et al. (2015), solved this problem through **residual connections** shortcut connections that bypass one or more layers, allowing gradients to flow directly through the network[95]. This simple but powerful idea made it possible to successfully train networks of 50, 101, and 152 layers, depths previously considered impossible to optimize. ResNet won the ILSVRC 2015 competition and its 50-layer variant, **ResNet-50**, became the most widely adopted backbone for downstream tasks including semantic segmentation(Figure 4.16)[95].

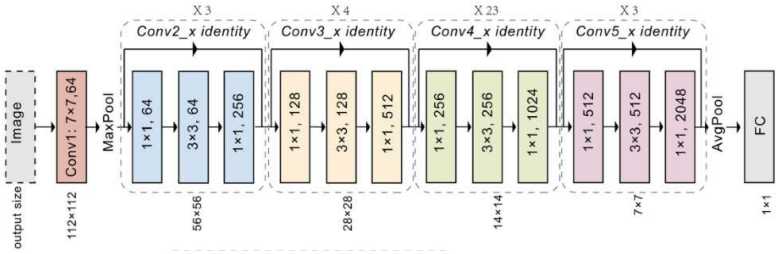


Figure 4.16: ResNet Architecture

Source:
https://www.ikomia.ai/blog/mastering-resnet-deep-learning-image-recognition

4.4.1.6 Limitations of Convolutional Neural Networks

Convolutional Neural Networks have proven highly effective at extracting local spatial features such as edges, textures, and object boundaries through their hierarchical convolutional layers. Their inductive biases, local connectivity, and weight sharing make them computationally efficient and well-suited for structured visual data.

However, the very design principle that makes CNNs powerful, the local receptive field, also represents their core structural limitation. Each neuron in a convolutional layer only processes a small neighborhood of the input, defined by the kernel size. Even when stacking many layers, the effective receptive field grows slowly and remains an approximation of true global context, meaning the network fundamentally struggles to model relationships between spatially distant regions of the image[96].

This limitation becomes critical in the context of semantic segmentation of high-resolution satellite imagery. Understanding that a group of pixels belongs to the class "urban area" or not requires reasoning about the surrounding context at a global scale: the presence of roads, buildings, and infrastructure spread across the entire image. A CNN, constrained by its local receptive field, cannot efficiently capture these long-range spatial dependencies.

Attempts to address these limitations, such as dilated convolutions, atrous spatial pyramid pooling (ASPP) [97], [98], pyramid pooling modules [99], and feature pyramid networks (FPN) [100], have partially extended the receptive field of CNNs. However, these solutions remain approximations.

These limitations create a clear need for an architecture that reasons over the entire image at once. This is precisely the problem that the Transformer architecture, through its self-attention mechanism, was originally designed to solve [101]. The following section introduces the Transformer and explains how its global reasoning capability directly addresses the structural weaknesses identified in convolutional networks.

4.4.2 Transformers

The Transformer architecture was introduced by Vaswani et al. (2017) in their seminal paper *Attention Is All You Need*. This model uses an encoder-decoder architecture but implements different attention mechanisms where both components are formed by stacking identical layers. Initially, the number of encoders and decoders was the same. However, this is no longer necessarily the case. In Transformers, an encoder is composed of a self-attention block followed by a feed-forward network. The decoder is also supplemented by an attention layer. Furthermore, Transformers introduced multihead attention mechanisms. These are very efficient but also very expensive to learn [101].

4.4.2.1 Attention

Attention is a method that reproduces cognitive attention. The objective of the attention technique is to focus on the key elements of an input and to neglect information that is not as relevant.

In the context of neural networks, an attention mechanism can be described as a function that performs a mapping between a query and a set of key-value pairs to an output. The query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key[101].

The following attention mechanisms are used in the Transformer architecture:

A. Scaled Dot-Product Attention The input consists of queries and keys of dimension d_k, and values of dimension d_v. The dot products of the query are computed with all keys, divided by $\sqrt{d_k}$, and a softmax function is applied to obtain the weights on the values[101].

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4.2}$$

The scaling factor $\frac{1}{\sqrt{d_k}}$ is essential: for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. Dividing by $\sqrt{d_k}$ counteracts this effect[101].

B. Multi-Head Attention Rather than performing a single attention function with d_{model}-dimensional keys, values and queries, Vaswani et al. found it beneficial to linearly project the queries, keys and values h times with different learned linear projections to d_k, d_k and d_v dimensions respectively. On each of these projected versions, the attention function is then performed in parallel, and the outputs are concatenated and once again projected[101]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{4.3}$$

$$\text{where} \quad \text{head}_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V) \tag{4.4}$$

where the projection matrices are $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

In the original work, $h = 8$ parallel attention heads were used, with $d_k = d_v = d_{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost remains similar to that of single-head attention with full dimensionality[101].

Multi-Head Attention allows the model to jointly attend to information from **different representation subspaces at different positions** ,something a single attention head cannot achieve, as averaging over a single attention head inhibits this capacity[101].

4.4.3 Vision Transformer (ViT)

Inspired by the success of Transformers in natural language processing, several attempts have been made to adapt this architecture to image analysis. The first successful adaptation was introduced by the Google Brain team, who presented the Vision Transformer (ViT) [6]. Similar to the sequence of word embeddings used when applying Transformers to text, ViT represents an input image as a sequence of fixed-size patches and predicts its class label using an encoder-only architecture. Figure 4.17 provides an overview of the Vision Transformer model.

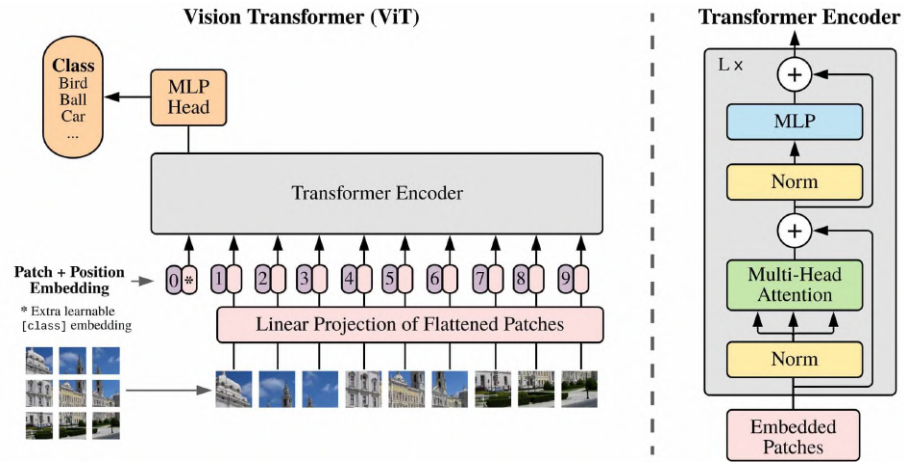


Figure 4.17: Vision Transformer (ViT) overview

Source: [6]

ViT architecture includes the following major components:

A. Image Patching & Linear Embedding : This stage converts a 2D image into a sequence of patch embeddings,The image is divided into fixed size and non-overlapping patches(typically 16×16), each treated as a token and converted into a 1D sequence format that the Transformer can process, exactly like word embeddings in NLP.

B. Positional Encoding : The Transformer processes all patch tokens in parallel and therefore has no inherent notion of spatial order and cannot tell where each patch came from in the original image. Therefore, the patch embeddings are added with positional encodings to retain the spatial information. In ViT, these positional encodings are learnable, meaning the model learns the best positional representation during training. Moreover, a special token, the [CLS] token, is appended at the beginning of the sequence to gather global information from all patches and pass it to the classification head for the final prediction.

C. Transformer Encoder The major advantage of attention in vision tasks is its ability to evaluate the relative importance of different parts of an image, favoring a global approach rather than a local one, unlike convolutions which are restricted to small local neighborhoods. Furthermore, it is not mandatory to use a full encoder-decoder structure. As proposed by Dosovitskiy et al., ViT adopts an encoder-only structure, consisting of the patch embeddings input followed by a succession of identical blocks, each block contains two main steps:

1. **Multi-Head Self-Attention (MSA):** Every single patch in the sequence is allowed to look at and interact with every other patch at the same time. The term multi-head means that this attention process is performed several times in parallel, each time focusing on different aspects of the relationships between patches, and the results are then combined together to form a richer and more complete representation .

2. **Feed-Forward Network (FFN):** After the attention step, each patch embedding passes independently through a small Feed-Forward Network made of two linear layers with a GELU activation function placed between them. Unlike ReLU which simply sets all negative values to zero, GELU behaves in a smoother and more probabilistic way, allowing small negative values to still contribute a little to the output, which helps the network learn more subtle and complex patterns.

Finally, after the last block, an MLP head is added to take the output of the `[CLS]` token and produce the final class prediction.

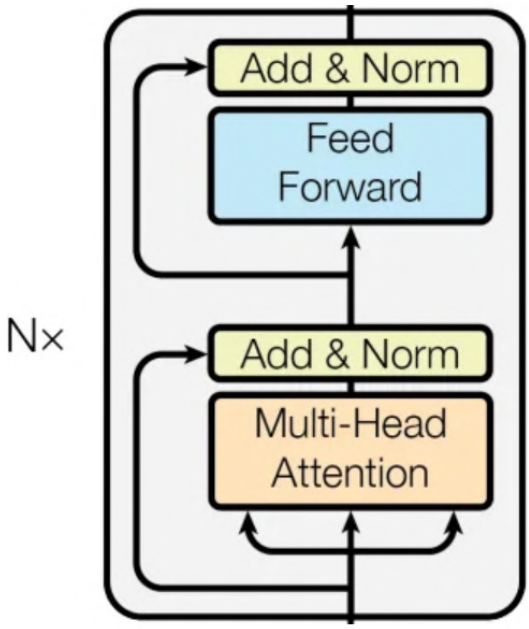


Figure 4.18: Encoder-Only Architecture
Source: https://cameronrwolfe.substack.com/i/95427942/encoder-only-architecture

D. Classification Head After the sequence of patch embeddings has passed through all the encoder blocks, the model needs to produce a final prediction. This is where the `[CLS]` token becomes essential. Throughout all the encoder layers, this token has been attending to every other patch in the sequence, collecting and summarizing the global information of the entire image into a single vector. This vector is then passed to the MLP head, which is a simple small network that maps this global representation to the final predicted class label. This design is elegant because it separates the job of understanding the image, done by the encoder, from the job of making a decision, done by the MLP head [6] [102] [103].

4.4.3.1 Limitations of Pure ViT

Despite its powerful global reasoning capability, the Vision Transformer presents several fundamental limitations that make it unsuitable as a standalone model for high-resolution satellite image segmentation .

A. Data-hungry ViTs require much larger datasets for training from scratch compared to CNNs and may not perform well with limited data unless pre-trained, thus presenting a higher risk of overfitting on smaller datasets . In the context of satellite image segmentation, where pixel-level annotated data is scarce and expensive to produce, this requirement represents a critical practical limitation [103].

B. Computational Complexity Due to the adoption of the Transformer's self-attention mechanism, ViT has a quadratic computational complexity $O(n^2)$ with respect to input size and is memory intensive. This can be prohibitively expensive

for high-resolution satellite images, where the number of patches n grows rapidly with image size [103].

C. Loss of Inductive Biases Unlike CNNs that have built-in assumptions about locality and translation invariance, ViTs need to learn these properties from data and may struggle with tasks where local features are crucial. This is particularly problematic for semantic segmentation of satellite images, where precise local boundary detection between semantic classes such as roads, buildings, and vegetation requires exactly the kind of local spatial sensitivity that CNNs naturally provide [103].

These three limitations data dependency, quadratic complexity, and loss of inductive biases collectively demonstrate that neither CNNs nor pure Vision Transformers alone are sufficient for high-resolution satellite image segmentation:

- **CNNs** provide strong local feature extraction but fail to capture global context.
- **Pure ViT** captures global context but loses local spatial precision, requires massive training data, and is computationally expensive at high resolutions.

The natural and scientifically motivated solution is therefore to combine both architectures into a **hybrid CNN-Transformer** model, which preserves the local inductive biases of CNNs while leveraging the global reasoning power of Transformers, which is precisely the approach adopted in this work.

4.4.4 Hybrid CNN-Transformer

The limitations of CNNs and pure vision transformers have motivated the development of hybrid architectures that combine both worlds. The main idea is CNNs are used to extract local spatial features, and Transformers are responsible for modeling global contextual dependencies, two complementary capabilities that neither architecture can provide alone.

This fusion strategy has proven effective across different domains and tasks. Chen et al. proposed TransUNet, which merits both Transformers and U-Net as a strong alternative for image segmentation. On one hand, the Transformer encodes tokenized image patches from a CNN feature map as the input sequence for extracting global contexts. On the other hand, the decoder combines the encoded features with high-resolution CNN feature maps to enable precise localization. [104]

In the specific context of remote sensing, Hwang et al. proposed SFA-Net, a hybrid architecture consisting of a CNN-based encoder and a transformer-based decoder, designed for the semantic segmentation of remote sensing images. Their work demonstrated that analyzing both local and global contexts through this hybrid design is essential for achieving robust performance on high-resolution satellite imagery and that the CNN-Transformer hybrid model is more suitable for efficiency than transformer-only approaches. [14] These works confirm that the hybrid CNN-Transformer paradigm is not only theoretically motivated but also empirically validated as the most effective strategy for semantic segmentation tasks, particularly

in the remote sensing domain. This directly justifies the hybrid architectural choice adopted in this work.

4.4.5 Hybrid Architectures

The rationale behind hybrid CNN-Transformer architectures is straightforward: CNNs excel at capturing local spatial details and fine-grained textures through convolutional operations, while Transformers are particularly effective at modelling long-range dependencies and global contextual relationships via self-attention. By combining these two paradigms, hybrid models can leverage the complementary strengths of both local and global representations, thereby improving the segmentation accuracy compared with that obtained through either CNN or Transformer-based networks alone[105].

TransUnet TransUNet, proposed by Chen et al. (2021), was the first framework to explore the potential of Transformers for image segmentation. It was motivated by a fundamental observation: pure CNN architectures such as U-Net fail to model long-range dependencies due to the intrinsic locality of convolution operations, while pure Transformers lack fine spatial details due to their treatment of images as 1D sequences. TransUNet was designed to merit both worlds by combining the local feature extraction power of CNNs with the global context modeling of Transformers [104]. The overall architecture of TransUNet is illustrated in Figure 4.19, showing a hybrid encoder-decoder design inspired by the U-shaped structure of U-Net.

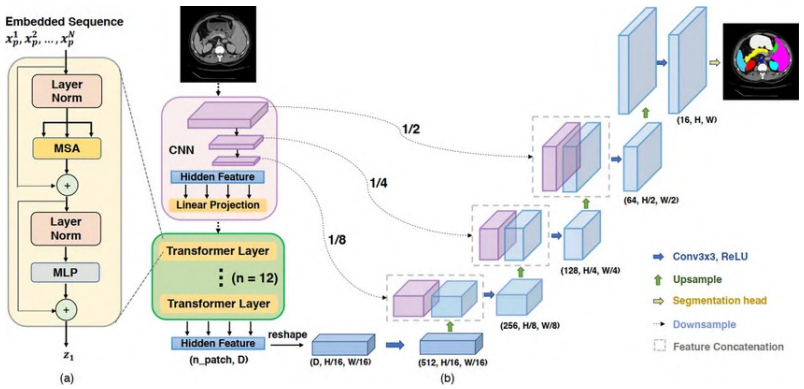


Figure 4.19: TransUNet architecture

Source: [104]
The architecture is composed of three main components:

- **A. CNN Feature Extraction:** A ResNet-50 backbone first processes the input image to generate hierarchical feature maps at multiple resolution scales. Unlike applying the Transformer directly to raw image patches, using CNN features as input allows the model to leverage rich low-level spatial details such as edges, textures, and boundaries, that the Transformer alone cannot preserve[104].

- **B. Transformer Encoding:** The CNN feature maps are tokenized into patch sequences through a linear projection combined with positional embeddings to retain spatial order. These patch sequences are then fed into a Transformer encoder consisting of multiple layers of Multi-head Self-Attention (MSA) and Feed-Forward Network (MLP) blocks. This allows the model to capture global contextual dependencies between all patches simultaneously, regardless of their spatial distance in the image[104].
- **C. Cascaded Upsampler and Skip Connections:** The encoded features are progressively upsampled back to the original image resolution through a Cascaded Upsampler (CUP), where each upsampling block doubles the spatial resolution. Critically, skip connections, inherited from the U-Net design, concatenate high-resolution CNN feature maps from the encoder with the upsampled Transformer features at each resolution level. This mechanism recovers the fine-grained spatial details that are lost during the Transformer encoding stage, enabling precise pixel-level localization [104].

4.4.6 Transfer learning

Transfer learning refers to an approach that enables knowledge learned from one task to be applied to a new (Figure 4.20), related task. Typically, networks are not trained from random initialization; instead, they start from a pre-trained model developed on large-scale datasets such as ImageNet [106].

Given this foundation, two strategies are widely used:

(1) using the pre-trained network as a fixed feature extractor, where early layers are frozen and only a new task-specific head is trained,

or (2) fine-tuning the network by adjusting some of the existing weights with a small learning rate [107].

This approach is especially valuable because it reduces training time, lowers the risk of overfitting, and enables strong performance even when labeled data is limited.

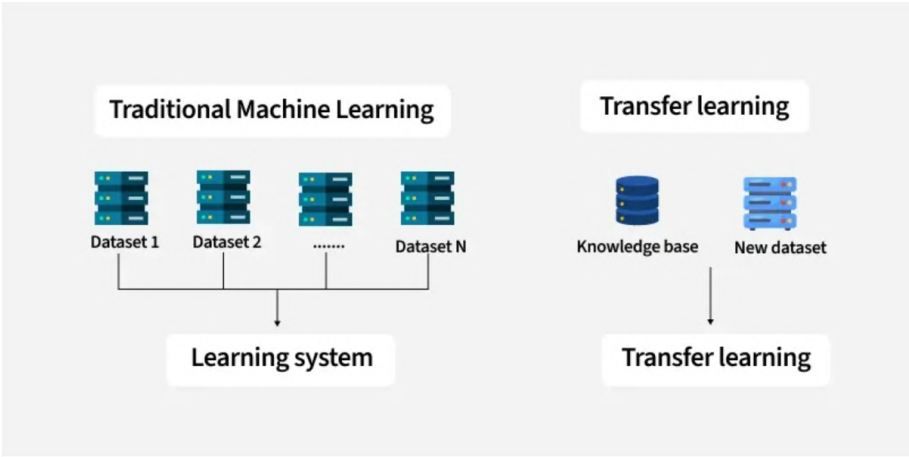


Figure 4.20: Transfer learning

Source: https://www.geeksforgeeks.org/machine-learning/transfer-learning-vs-fine-tuning-vs-multitask-learning-vs-federated-learning/

4.4.7 Overfitting and Regularization

4.4.7.1 Overfitting

Overfitting is a common issue in deep learning, where a model becomes too complex and fits the training data too well, resulting in poor generalization performance on unseen data. As described by Mungoli, overfitting occurs when a model learns the noise in the training data instead of the underlying patterns, leading to a model that performs well on the training data but poorly on new unseen data [108].

Overfitting can be detected by monitoring the training and validation loss during training. If the training loss decreases while the validation loss increases, it is a clear indication that the model is overfitting to the training data, as illustrated in Figure 4.21.

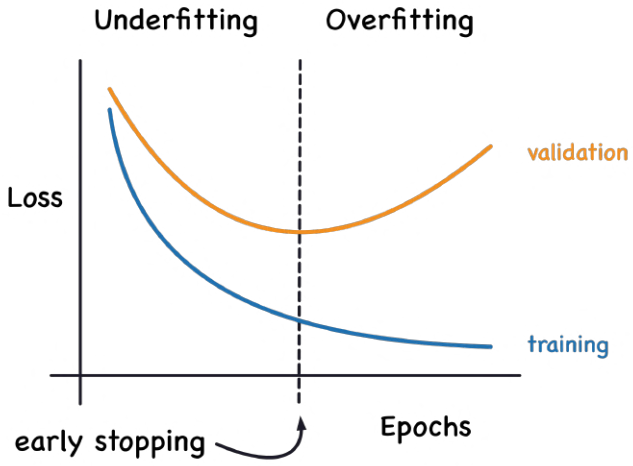


Figure 4.21: Illustration of overfitting.
Source: https://www.kaggle.com/code/ryanholbrook/overfitting-and-underfitting

4.4.7.2 Regularization Techniques

To address overfitting during the training process, it is important to apply preventive measures. Several techniques are commonly used [108]:

- **Data Augmentation:** Generating additional training data by applying random transformations to the existing training samples, such as flipping, rotation, and scaling. This provides the model with more varied examples and reduces its tendency to memorize the training data.
- **Dropout:** Randomly deactivating a portion of neurons during each training iteration, forcing the model to learn more robust and distributed representations rather than relying on a single set of neurons.
- **Hyperparameter Adjustment:** Carefully tuning the model hyperparameters, including the number of layers, type of activation function, learning rate, optimizer (SGD, Adam, etc.), batch size, and number of epochs, in order to find the best configuration that balances model complexity and generalization.

- **Early Stopping:** Terminating the training process when the model performance on the validation set stops improving, preventing the model from continuing to fit the noise in the training data.

4.4.8 Evaluation Metrics

The performance of semantic segmentation models is typically evaluated using standardized metrics widely accepted in remote sensing and computer vision. The key metrics employed in this work are defined as follows:

Intersection over Union (IoU) / Jaccard Index IoU is one of the most widely adopted metrics in semantic segmentation. It quantifies the overlap between the predicted segmentation map A and the ground truth B:

$$\text{IoU} = J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4.5}$$

where the result ranges from 0 (no overlap) to 1 (perfect match) [60]. IoU is robust to class imbalance and directly reflects segmentation quality at the region level.

Mean IoU (mIoU) Mean IoU averages the IoU score across all semantic classes:

$$\text{mIoU} = \frac{1}{K+1} \sum_{i=0}^{K} \text{IoU}_i. \tag{4.6}$$

This metric serves as the primary evaluation criterion for benchmarks such as the LoveDA dataset [63].

Precision, Recall, and F1-Score Precision and recall evaluate the trade-off between false positives and false negatives for each class:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{4.7}$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively. The F1-score combines both into a single harmonic mean:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4.8}$$

These metrics are particularly useful for evaluating performance on imbalanced classes such as ”building” or ”road” in satellite imagery [60].

Dice Coefficient The Dice coefficient measures twice the overlap area divided by the total number of pixels in both prediction and ground truth:

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}. \tag{4.9}$$

For binary segmentation, Dice is mathematically equivalent to the F1-score [60]. It is widely used in medical image analysis and is equally applicable to remote sensing tasks where precise boundary delineation is critical.

4.5 Conclusion

This chapter has reviewed the core theoretical concepts of deep learning necessary to address the semantic segmentation of satellite imagery. We have shown that while CNNs excel at capturing local spatial patterns, they struggle with long-range dependencies; conversely, Vision Transformers provide global reasoning but require substantial data and computational resources. Hybrid CNN-Transformer architectures emerge as a scientifically motivated solution, leveraging the complementary strengths of both paradigms. We have also highlighted the importance of transfer learning and regularization strategies to address data scarcity and overfitting,critical challenges in remote sensing applications.

The metrics and challenges discussed herein provide the evaluation framework for our experimental work. In the next chapter, we present the proposed hybrid architecture, detail its implementation , and analyze the results obtained followed by a comprehensive discussion of performance and limitations.

Chapter 5

Implementation and Discussion of the Results

5.1 Introduction

This chapter presents how we implemented our research, including training configurations and results found in the proposed research. We built on what we discussed in previous chapters, which covered the theory and how we prepared the data. The primary objective here is to see and evaluate how effective the hybrid CNN–Transformer architecture is compared to a convolutional neural network (CNN) for the semantic segmentation of high-resolution satellite imagery.

In this experiment, we used the LoveDA dataset with 3 channel RGB imagery data to classify different and complex land-cover types, such as buildings and areas with vegetation.

5.2 Development environments and tools

5.2.1 kaggle

Kaggle is a famous platform where data scientists do competitions and work on analytics projects, launched in 2010 [109]. Data scientists and researchers go together to Kaggle to work on data analysis and predictive modeling. Companies and organizations share their data and the problems they are trying to solve. From there, practitioners use these data to build machine learning models and make predictions with them and submit their found results to see how they do compared to others on the Kaggle leaderboard [110].

Figure 5.1: logo of kaggle

5.2.1.1 Kaggle Notebooks

Kaggle Notebooks provide versioned computational environments powered by cloud-based Docker containers. They allow users to write and execute code in both R and Python[111]. Kaggle offers robust computational resources to help you create baseline solutions, but there are limits on how much you can use them; you get a certain amount of power each week.

Here are the details in the table below:

Notebook type	CPU cores	Memory	Number of notebooks that can be run at a time	Weekly quota
CPU	4	16 GB	10	Unlimited
GPU	2	13 GB	2	30 hours
TPU	4	16 GB	2	30 hours

Figure 5.2: Kaggle Notebook computational resources and usage quotas [111]

Kaggle Notebooks are really useful because they provide an interactive interface. and it comes with pre-installed machine learning libraries such as TensorFlow, PyTorch, and Scikit-Learn.

5.2.1.2 Datasets and Models

The Meta Kaggle Code dataset offers a vast archive of publicly shared notebooks; these notebooks have about 5.9 million scripts with their complete source code and metadata (such as timestamps and author information). It has all this from 2015 to now, and it is around 300 gigabytes; it offers an excellent resource for learning and working with data science in the world. [112].

5.2.1.3 Competitions and Benchmarking

Kaggle challenges task participants to develop models that can solve problems using certain datasets. When you submit your model, it gets scored, and you can see how you are doing on a leaderboard that everyone can see. To avoid overfitting, they use some secret test data to figure out the final rankings. Participants who make the best models get prizes, and sometimes they have to give the company that runs the challenge the rights to the code they wrote for the model [113].

5.2.2 Python

Python is an open-source language. It was made by Guido van Rossum. It is very easy to learn and very used by students in learning and also in big projects [114]. Python is an effective language because it is simple to read and write and has useful tools that make it easy to work with data and make programs that can do many things, making it an ideal tool in different platforms and environments [115].

Figure 5.3: Logo of python

5.2.3 The libraries used

In this project, several libraries were used for data manipulation, creating the segmentation hybrid model, training, evaluation, and visualization of the results. The main libraries used are:

5.2.3.1 TensorFlow

It was originally made by Google as part of the Google Brain project, and then it became an open-source framework [116]. It is widely useful in different scientific fields because it can do a lot of complicated math with things called ”tensors.” These tensors are like objects that can do some math and then give us an answer. [117].

5.2.3.2 Keras (integrated into TensorFlow)

Keras is a high-level deep learning API made to be easy to use and understand; it helps you find errors quickly and makes your code look clean and easy to read. It gives you building blocks; these blocks make it easy to work on. It is also easy to reuse [118].

5.2.3.3 NumPy (Numerical Python)

NumPy is an important package for doing numerical computing in Python. It has a team of people who help make it better; with over 700 contributors working on it, NumPy is great for working with sets of numbers like N-dimensional arrays and matrices. It also has lots of math functions that make it easy to work with these numbers [117].

5.2.3.4 pandas (Python Data Analysis Library)

Pandas is an important tool in data science. It is often used with NumPy and Matplotlib for tasks like data manipulation and cleaning. The Pandas community

has over 1,200 contributors who help support it; it provides high-performance, easy-to-use data structures and analysis tools [117].

5.2.3.5 matplotlib

Matplotlib is a great library for Python that helps in making good-looking graphs and charts. It has a lot of people working on it, around 700 people, who make sure it keeps getting better. Matplotlib is an important tool for the community that works with data. In the data science field, the Matplotlib library is used in visualizations and data plots [117].

5.2.3.6 Scikit-learn (sklearn)

Scikit-learn is an open-source library. This library is built using popular Python tools like NumPy, SciPy, and Matplotlib. These tools help Scikit-learn provide easy-to-reuse functions for different data analysis tasks, such as making predictions and data analysis [119].

5.2.3.7 seaborn

Seaborn is a Python library for data visualization. It is made using matplotlib. It is really good at visualizing graphs and easy to understand with informative statistics [120].

5.3 Database used

5.3.1 LoveDA Dataset

5.3.1.1 LoveDA Dataset Overview

The LoveDA dataset was made using high spatial resolution imagery collected in July 2016 from three different cities in China: Nanjing, Changzhou, and Wuhan, where this imagery was taken from a height of 0.3 meters above the ground. The LoveDA dataset covers an area of 536.15 km^2 and consists of 5,987 images containing 166,768 marked objects. The LoveDA dataset is special because it has pictures from two kinds of places: rural and urban. There are 2,713 images of the urban and 3,274 images of the rural in the dataset [63].

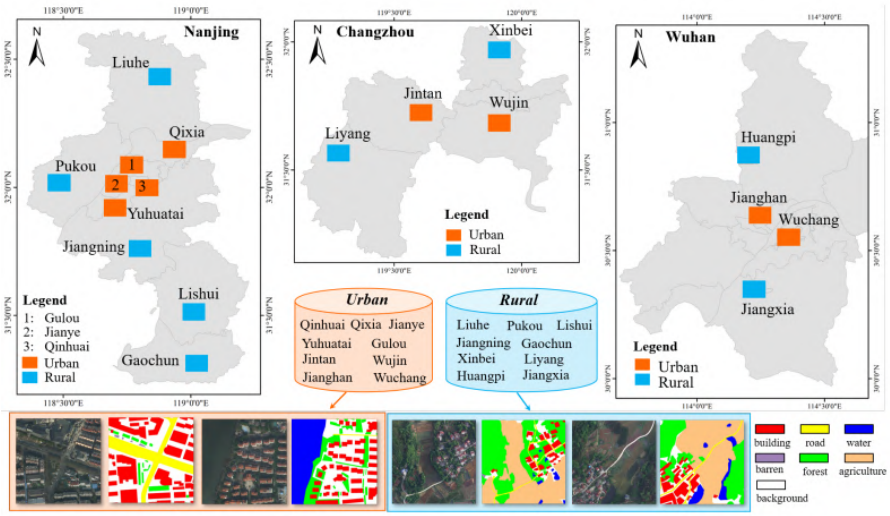


Figure 5.4: Overview of the dataset distribution [63]

The details of the dataset are as follows:

- **Image Dimensions & Channels:** The images are high-resolution RGB with a size of 1024×1024 pixels.
- **Storage Size:** The whole dataset takes around 9.58 GB of space. This includes masks and annotations. The images alone take about 5.8 GB.
- **Data Splits:** The dataset is divided into three parts: training, validation, and testing. This helps to ensure robust model evaluation:
  - **Training Set:** 2,522 images (approx. 2.4 GB)
  - **Validation Set:** 1,669 images (approx. 1.7 GB)
  - **Test Set:** 1,796 images (approx. 1.8 GB)

5.3.1.2 Semantic Categories

The dataset is sorted into seven groups, and we also have a background group that is called "no-data." This background group is assigned the number 0:

1. Background (No-data)
2. Building
3. Road
4. Water
5. Barren
6. Forest
7. Agriculture

5.3.2 LandCover Dataset

5.3.2.1 LandCover Dataset Overview

The LandCover.ai (Land Cover Analysis for Aerial Imagery) The LandCover.ai dataset is a large-scale collection specifically designed to push the boundaries of semantic segmentation in aerial mapping [121]. It covers an extensive area of 216.27 km^2 within Poland, using data provided by the Polish Head Office of Geodesy and Cartography. The imagery is highly detailed, featuring spatial resolutions of 25 cm and 50 cm per pixel [121].

(a) Geographical locations of ortho-images (b) Diversity of environmental conditions

Figure 5.5: Geographical distribution (left) and representative aerial samples (right) of the LandCover.ai dataset [122].

The technical specifications of the dataset are as follows:

- **Image Dimensions & Channels:** The dataset consists of high-resolution RGB ortho-images. In this study, these were processed into tiles of 256×256 pixels to optimize GPU memory during training.
- **Storage Size:** The dataset occupies approximately 2.1 GB of disk space when extracted, containing both the raw imagery and the corresponding labeled masks.
- **Data Splits:** To ensure a fair and robust evaluation, the data was partitioned into three subsets following the original author's split logic to prevent data leakage:
  - **Training Set:** 7,470 images (approx. 70%)
  - **Validation Set:** 1,602 images (approx. 15%)
  - **Test Set:** 1,602 images (approx. 15%)

5.3.2.2 Semantic Categories

The dataset is structured into five distinct semantic classes. Each pixel is assigned a label ranging from 0 to 4, representing the following land cover categories [121]:

1. Background (0)
2. Building (1)
3. Woodland (2)
4. Water (3)
5. Road (4)

5.4 Proposed Network Architectures

5.4.1 Model 1:U-Net with ResNet50 Encoder (CNN)

5.4.1.1 Application on LoveDA

1. **Architecture:**

   The model follows the classic encoder–decoder paradigm of the **U-Net** architecture. The encoder is the **ResNet50** backbone, whose feature maps are extracted at four resolution levels and passed to the decoder via skip connections. The bottleneck output is progressively upsampled through four decoder blocks, each performing bilinear upsampling by a factor of 2, concatenating the corresponding encoder skip connection, and applying two 3×3 convolutional layers with Batch Normalization and ReLU activation.

   A `Dropout` layer with rate 0.2 is inserted before the final output layer to reduce overfitting. A 1×1 convolutional layer followed by **Softmax** activation produces pixel-wise class probabilities over the 7 land-cover classes.

2. **Data Loading:**

   Images are loaded from the LoveDA directory structure, which organises Urban and Rural subsets under separate `Train`, `Val`, and `Test` folders provided by the benchmark. Each image is loaded in RGB format using OpenCV, and its corresponding mask is loaded in grayscale. Mask pixel values are shifted by -1 to remap the original label range $[1, 7]$ to $[0, 6]$, and clipped to ensure validity.

   A custom **`LoveDAGenerator`** class, derived from `tf.keras.utils.Sequence`, handles dynamic and memory-efficient loading of image-mask pairs during training, reading images batch by batch without loading the full dataset into memory.

   An identical baseline U-Net architecture with a ResNet50 encoder is utilized for the LoveDA dataset, matching the structural framework illustrated in Figure 5.7. However, while the LandCover.ai configuration optimizes for 5 target

categories, the final classification layer for the LoveDA pipeline is adjusted to output 7 distinct semantic classes. Specifically, the network is trained to segment background, building, road, water, barren, forest, and agricultural land, compared to the background, building, woodland, road, and water classes defined in the LandCover.ai dataset.

```
class LoveDAGenerator(Sequence):
    def __init__(self, img_dirs, mask_dirs, batch_size=8, img_size
    =512,
                 shuffle=True, training=True, multi_scale=False):
        self.imgs, self.masks = [], []
        for img_dir, mask_dir in zip(img_dirs, mask_dirs):
            self.imgs.extend(sorted(glob(os.path.join(img_dir, "*.
    png"))))
            self.masks.extend(sorted(glob(os.path.join(mask_dir, "
    *.png"))))
        self.bs          = batch_size
        self.img_size    = img_size
        self.shuffle     = shuffle
        self.training    = training
        self.multi_scale = multi_scale
        self.idx         = np.arange(len(self.imgs))
        self.scales      = [0.5, 0.75, 1.0, 1.25, 1.5, 1.75]
        if shuffle:
            np.random.shuffle(self.idx)
```

Listing 5.1: The `LoveDAGenerator` class used for dynamic and memory-efficient loading of LoveDA image-mask pairs during training.

3. **Data Augmentation:**

   After loading the dataset, a significant class imbalance at the pixel level was observed, with **Background** and **Agricultural** classes heavily over-represented, while **Barren** and **Water** classes are much less frequent, as illustrated in Figure 5.6.

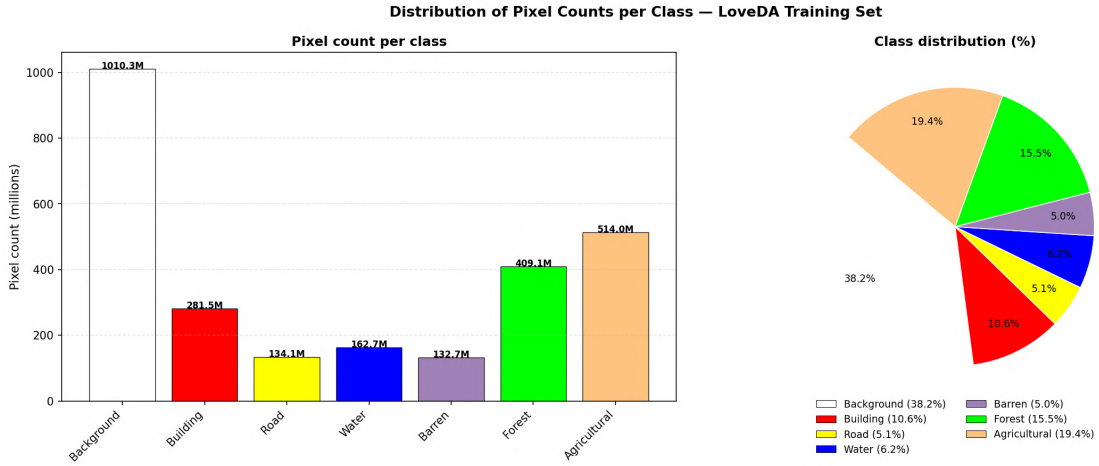


Figure 5.6: Distribution of pixel counts per class in the LoveDA training dataset.

To improve model generalisation without introducing class-specific oversampling (which caused severe overfitting in our preliminary experiments), we applied a **uniform augmentation strategy** to all training images through the custom generator. The augmentations applied are:

- **Multi-scale resizing**: each image-mask pair is randomly rescaled using a scale factor drawn from $\{0.5, 0.75, 1.0, 1.25, 1.5, 1.75\}$ with probability 0.5, using bilinear interpolation for the image and nearest-neighbor interpolation for the mask to preserve label integrity.
- **Random crop or resize**: if the scaled image exceeds 512×512, a random spatial crop is extracted; Otherwise, the image is directly resized to 512×512.
- **Horizontal and vertical flipping**: each applied independently with probability 0.75.
- **Random rotation**: the image and mask are rotated together by a randomly chosen multiple of 90°, also with probability 0.75.

All mask transformations use nearest-neighbor interpolation to preserve integer label values. **No augmentation is applied at validation time**; the validation generator applies only a deterministic resize to 512×512 pixels to ensure reproducible and comparable evaluation scores.

```
def _augment(self, img, mask):
    # Multi-scale resize (article) - train seulement
    if self.multi_scale and np.random.rand() > 0.5:
        s  = np.random.choice(self.scales)
        nh = int(img.shape[0] * s)
        nw = int(img.shape[1] * s)
        img  = cv2.resize(img,  (nw, nh), interpolation=cv2.
    INTER_LINEAR)
        mask = cv2.resize(mask, (nw, nh), interpolation=cv2.
    INTER_NEAREST)

    # Random crop ou resize vers IMG_SIZE
    h, w = img.shape[:2]
    if h > self.img_size or w > self.img_size:
        i = np.random.randint(0, max(1, h - self.img_size + 1))
        j = np.random.randint(0, max(1, w - self.img_size + 1))
        img  = img[i:i+self.img_size,   j:j+self.img_size]
        mask = mask[i:i+self.img_size,  j:j+self.img_size]
    else:
        img  = cv2.resize(img,  (self.img_size, self.img_size))
        mask = cv2.resize(mask, (self.img_size, self.img_size),
    interpolation=cv2.INTER_NEAREST)

    # Flips & rotation (p=0.75 comme article)
    if np.random.rand() > 0.25:
        img, mask = np.fliplr(img).copy(), np.fliplr(mask).copy()
    if np.random.rand() > 0.25:
        img, mask = np.flipud(img).copy(), np.flipud(mask).copy()
```

```
    if np.random.rand() > 0.25:
        k = np.random.randint(0, 4)
        img, mask = np.rot90(img, k).copy(), np.rot90(mask, k).
    copy()

    return img, mask
```

Listing 5.2: Python implementation of the multi-scale transformation, spatial cropping, and geometric augmentation pipeline inside the data generator.

4. **Data Split:**

   The LoveDA dataset provides an official split into separate `Train`, `Val`, and `Test` folders. We used the official Train split (Urban + Rural) for training and the official Val split for validation. The Test set contains no ground-truth masks and is used only for qualitative visual predictions. The final distribution is summarised in Table 5.1.

   Table 5.1: Distribution of LoveDA data into Train / Val / Test subsets.

Subset	Urban	Rural	Total
Train	1156	1366	**2522**
Validation	677	992	**1669**
Test	–	–	**1796**

5. **Data Normalization:**

   Images are normalised by dividing pixel values by 255, mapping them to the range $[0,\ 1]$, and converted to `float32` tensors. Masks are converted to one-hot encoded tensors of shape $512 \times 512 \times 7$ using `to_categorical` from Keras, as required for training with the categorical composite loss function.

```
    img = img.astype(np.float32) / 255.0
    mask = np.clip(mask.astype(np.int32) - 1, 0, NUM_CLASSES - 1)
    mask = to_categorical(mask, num_classes=NUM_CLASSES)
    X.append(img); y.append(mask)

if len(X) == 0:
    return self[0]
return np.array(X, dtype=np.float32), np.array(y, dtype=np.float32
    )
```

Listing 5.3: The normalization and one-hot encoding applied to LoveDA images and masks before training.

5.4.1.2 Application on Land Cover.ai

This section details the implementation workflow for the baseline U-Net model built with a ResNet50 backbone, applied to the LandCover.ai dataset for high-resolution semantic segmentation:

1. **Architecture:** Our proposed model follows an encoder-decoder framework tailored for high-resolution satellite imagery. For the encoder, we use a pre-trained ResNet50 backbone that processes $512 \times 512 \times 3$ RGB images, leveraging transfer learning to instantly recognize basic textures and shapes across five extraction stages. The compressed outputs meet at the bottleneck, the deepest part of the network, which captures abstract, highly dense semantic data with a broad global view. The decoder then mirrors this process to upscale the maps back to the original size using convolutions and Batch Normalization. Crucially, skip connections pass sharp, early-stage spatial details directly from the encoder to the decoder to recover fine boundaries like roads and building corners. Finally, to prevent overfitting on the LandCover.ai dataset, the features pass through a 0.2 Dropout layer before a final convolution with a Softmax activation outputs a precise pixel-level prediction tensor mapped across the 5 target land-cover classes.

   See Figure 5.7.

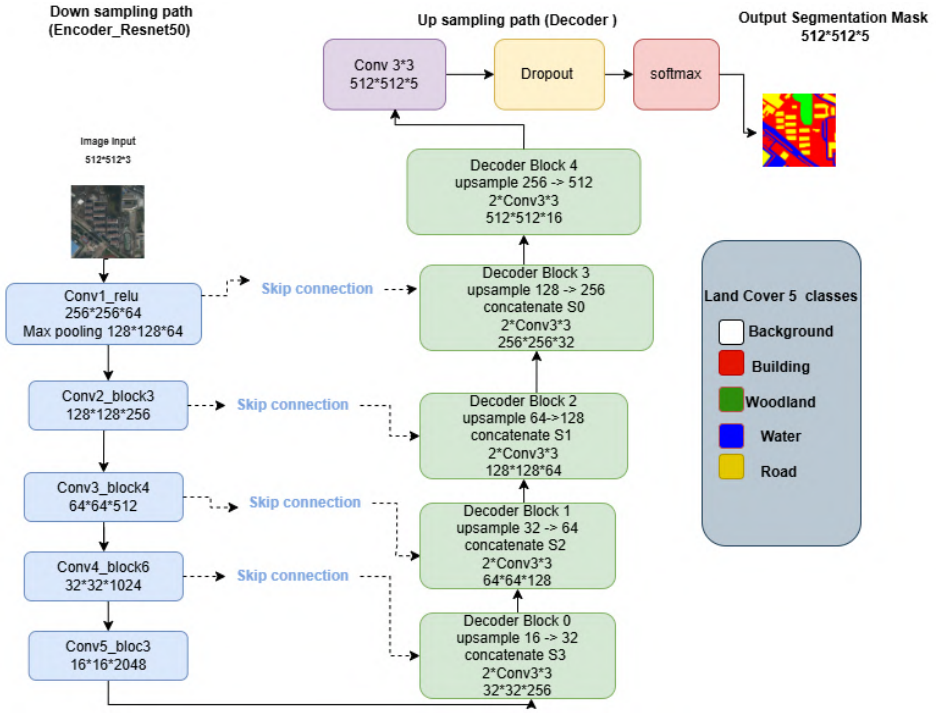


Figure 5.7: Architecture of the U-Net ResNet50 model applied to the LandCover.ai dataset.

2. **Data Loading:** Data loading is handled by a custom `LandCoverGenerator` class built on Keras's `Sequence` utility. This custom class serves as a thread-safe data pipeline that streams batches of images dynamically from the disk to the GPU, preventing system memory overloads. It pairs the 3-channel RGB aerial images (`.jpg`) with their matching 1-channel 8-bit grayscale ground-truth masks (`.png`). Every image is read at its native patch size of 512×512 **pixels** based on the `IMG_SIZE` setting. The code also includes an error check to skip missing files, along with a nearest-neighbor interpolation guard

(`cv2.INTER_NEAREST`) to make sure the categorical mask borders stay clean and free of artificial values during any rare resizing anomalies.

```
        # --- Processing and Disk Loading Loop inside __getitem__ ---
        for i in ids:
            img  = cv2.cvtColor(cv2.imread(self.img_paths[i]), cv2.COLOR_BGR2RGB)
            mask = cv2.imread(self.mask_paths[i], cv2.IMREAD_GRAYSCALE)

            if img is None or mask is None:
                print(f" Cannot read: {self.img_paths[i]}")
                continue

            # Resize fallback only if a tile is not exactly standard dimensions
            if img.shape[:2] != (self.img_size, self.img_size):
                img  = cv2.resize(img,  (self.img_size, self.img_size))
                mask = cv2.resize(mask, (self.img_size, self.img_size),
                                  interpolation=cv2.INTER_NEAREST)
```

Listing 5.4: Disk I/O Parsing and Structural Image Loading Framework within the LandCoverGenerator class

3. **Data Augmentation:** To stop the model from simply memorizing the training images, data augmentation runs live inside the data generator. These changes happen randomly, with a 75% chance of applying any given transformation to a batch. The setup includes horizontal flips using `np.fliplr`, vertical flips using `np.flipud`, and discrete spatial rotations (90°, 180°, or 270°) using `np.rot90`. These transformations are applied to both the aerial image and its corresponding mask at the exact same time using matching random seeds to ensure strict spatial alignment. Augmentations are limited strictly to the training phase; the validation data remains untouched to keep performance tracking clean and fair.

```
    def _augment(self, img, mask):
        if np.random.rand() > 0.25:
            img, mask = np.fliplr(img).copy(), np.fliplr(mask).copy()
        if np.random.rand() > 0.25:
            img, mask = np.flipud(img).copy(), np.flipud(mask).copy()
        if np.random.rand() > 0.25:
            k = np.random.randint(1, 4)   # 90/180/270 degrees
```

```
8             img, mask = np.rot90(img, k).copy(), np.rot90(mask, k)
  .copy()
9         return img, mask
```

Listing 5.5: On-the-Fly Geometric Data Augmentation Subroutine

See Figure 5.8.

(a) Original Image (b) Original Mask (c) Augmented Img (d) Augmented Msk

Figure 5.8: Examples of augmented training images and their corresponding ground-truth masks for the LandCover.ai dataset.

4. **Data Split:** Instead of using a random percentage-based split in Python, this pipeline follows the official, standard benchmark split provided directly by the LandCover.ai dataset creators. The splits are loaded using the reference text files inside the `raw_data/` directory: `train.txt` is parsed to build the primary training pool handled by `train_gen`, while `val.txt` forms the independent validation pool handled by `val_gen` to manage learning rate reductions, trigger early stopping, and guide hyperparameter choices. Sticking to this standard partition ensures our results can be directly and fairly evaluated against existing academic publications. The categorical configuration of classes used across these splits is summarized in Table 5.2.

Table 5.2: Target class configuration, indexing, and hex/RGB color mappings for the LandCover.ai dataset partitions.

Class ID	Class Name	Target Mask Color Label (RGB)
0	Background	White [255, 255, 255]
1	Building	Red [255, 0, 0]
2	Woodland	Green [0, 255, 0]
3	Water	Blue [0, 0, 255]
4	Road	Yellow [255, 255, 0]

5. **Data Normalization:** Before entering the neural network, the images and masks are processed to fit the mathematical requirements of the model. The original 8-bit RGB channels containing pixel values from 0 to 255 are divided by 255.0 to scale them down into a clean floating-point range between 0.0 and 1.0. Simultaneously, to match the model's 5-channel output, the

grayscale ground-truth labels (clamped between 0 and 4 to handle edge boundaries) are transformed into one-hot encoded binary matrices using Keras's `to_categorical` tool. This reshapes the target data into a (Batch Size, 512, 512, 5) tensor, where each slice along the last dimension represents one of our 5 specific land-cover targets. This prepares the data perfectly for categorical focal and dice loss optimizations.

```
            # --- Rescaling and One-Hot Encoding inside __getitem__ ---
            img  = img.astype(np.float32) / 255.0
            mask = np.clip(mask.astype(np.int32), 0, NUM_CLASSES - 1)
            mask = to_categorical(mask, num_classes=NUM_CLASSES)

            X.append(img)
            y.append(mask)
```

Listing 5.6: Batch Pixel Scaling and Target One-Hot Encoding Operations

5.4.2 Model 2: TransUNet with ResNet50V2 Encoder (Hybrid CNN-Transformer)

5.4.2.1 Application on LoveDA

1. **Architecture:** The second model is a **TransUNet** architecture, which represents a hybrid CNN–Transformer approach combining the local feature extraction capability of convolutional neural networks with the global context modelling power of Transformers. The architecture consists of three main components, as illustrated in Figure 5.9:

   - **CNN Encoder (ResNet50V2)**: The encoder backbone is a **ResNet50V2** pre-trained on ImageNet, which extracts multi-scale feature maps at resolution levels used as skip connections for the decoder.
   - **Transformer Bottleneck**: The bottleneck feature map (32×32) is first flattened into a sequence of $n = 32 \times 32 = 1024$ tokens, then projected to a $d = 512$ dimensional embedding space via a linear projection layer. A learnable **Positional Encoding (PE)** layer is added to inject spatial information. The sequence is then passed through **12 Transformer blocks**, each consisting of:
     - Layer Normalisation ($\epsilon = 10^{-6}$)
     - Multi-Head Self-Attention (8 heads) with stochastic Dropout increasing progressively from 0.30 to 0.40 over the 12 blocks
     - Feed-Forward Network with GELU activation (hidden dimension = 2048)
     - Residual connections after each sub-layer
   - **CNN Decoder**: The Transformer output is reshaped back to spatial dimensions ($32 \times 32 \times 512$) and progressively upsampled through 4 de-

coder stages. Each stage performs bilinear upsampling by $\times 2$, concatenates the corresponding ResNet50V2 skip connection, and applies two convolutional layers with Batch Normalisation, ReLU activation, and `SpatialDropout2D` (rate $= 0.4$). The decoder filter sizes are $\{256, 128, 64, 32\}$ respectively. A final 1×1 convolutional layer followed by **Softmax** activation produces pixel-wise predictions over the 7 land-cover classes (see Figure 4.19).

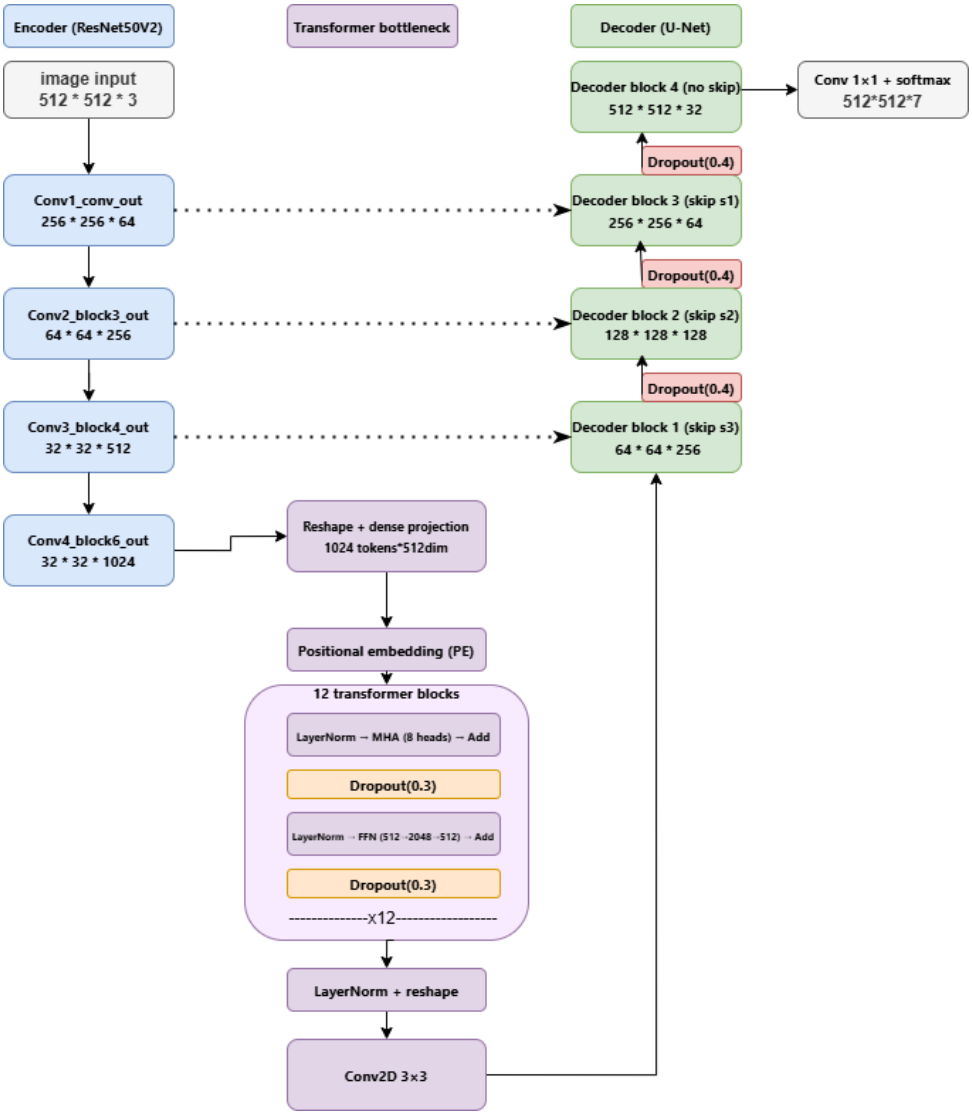


Figure 5.9: Architecture diagram of the hybrid TransUNet model applied to the LoveDa dataset.

2. **Data Loading:**

   Images are loaded using the `tf.data` pipeline for efficient and memory-safe data ingestion. Each image path is read using `tf.io.read_file`, decoded as a 3-channel PNG using `tf.image.decode_png`, and resized to 512×512 pixels. The image pixels are then preprocessed using the `preprocess_input` function

from `ResNet50V2`, which normalises the values to the range $[-1, 1]$ according to the ImageNet statistics. Mask files are loaded in grayscale (1 channel), resized using nearest-neighbor interpolation, and clipped to remap label values from $[1, 7]$ to $[0, 6]$. The entire pipeline leverages `tf.data.AUTOTUNE` for parallel prefetching and processing.

```
def load_and_preprocess(pi, pm):
    img = preprocess_input(tf.cast(tf.image.resize(
        tf.image.decode_png(tf.io.read_file(pi), channels=3),
        [IMG_SIZE, IMG_SIZE]), tf.float32))
    mask = tf.cast(tf.squeeze(tf.image.resize(
        tf.image.decode_png(tf.io.read_file(pm), channels=1),
        [IMG_SIZE, IMG_SIZE], method='nearest'), axis=-1), tf.
    int32)
    mask = tf.clip_by_value(mask - 1, 0, NUM_CLASSES - 1)
    return img, mask
```

Listing 5.7: the `load_and_preprocess` function used in the TransUNet `tf.data` pipeline.

3. **Data Augmentation:** A rich augmentation strategy is applied exclusively during training to improve generalisation and reduce overfitting. The augmentation pipeline, implemented as a mapped `tf.data` transformation, includes the following operations applied to each image–mask pair:

   - **Multi-scale random resizing**: a random scale factor is uniformly drawn from $[0.75,\ 1.5]$. The image is resized using bilinear interpolation, while the mask uses nearest-neighbor interpolation to preserve label integrity.
   - **Random crop or REFLECT padding**: if the rescaled image is larger than 512×512, a random spatial crop of size 512×512 is extracted from both image and mask simultaneously. If smaller, symmetric `REFLECT` padding is applied to avoid border artefacts, followed by a deterministic centre crop.
   - **Random horizontal flip**: applied with probability 0.5 consistently to both image and mask.
   - **Random vertical flip**: applied with probability 0.5 consistently to both image and mask.
   - **Random 90° rotation**: the image and mask are rotated together by a randomly chosen multiple of 90° ($k \in \{0, 1, 2, 3\}$).
   - **Color jitter** (image only):
     - Random brightness: $\Delta \in [-0.3,\ 0.3]$
     - Random contrast: factor $\in [0.7,\ 1.3]$
     - Random saturation: factor $\in [0.7,\ 1.3]$
     - Random hue: $\Delta \in [-0.05,\ 0.05]$

     Final pixel values are clipped to $[-1.0,\ 1.0]$.

No augmentation is applied at validation time

The validation pipeline applies only the deterministic `load_and_preprocess` function (resize to 512 × 512 + normalisation), ensuring reproducible and unbiased evaluation scores.

4. **Data Split:**

   - **Official LoveDA Split**

     In our first experiment, we used the official LoveDA benchmark split. Since the Test set provided by the benchmark contains **no ground-truth masks**, only the labelled Train and Validation subsets are available for quantitative evaluation. The distribution of labelled images is summarised in Table 5.3.

Table 5.3: Official LoveDA labelled split used in TransUNet experiment.

Subset	Images	% of Labelled	Usage
Train	2,522	60.2%	Model training
Validation	1,669	39.8%	Tuning / EarlyStopping / Evaluation
Total labelled	**4,191**	**100%**	

After training with the official split, two major limitations were identified that jointly explain the relatively modest validation mIoU of **44.50%**: (1) **data insufficiency**, the official split allocates only 60% of the available labelled images to training, and Transformer-based architectures require significantly larger datasets to learn meaningful token relationships; (2) **GPU memory and session constraints** on the Kaggle free platform (2× Tesla T4, 16 GB VRAM each), which limited the batch size to 8 and the total training time to approximately 10–11 hours.

These limitations motivated the design of the **proposed 80/20 custom split**, which provides **+33% more training images** (3,352 vs 2,522) while keeping the validation set large enough (839 images) for reliable EarlyStopping monitoring.

   - **Proposed 80/20 Balanced Custom Split**

     After analysing the dataset statistics and conducting several experiments, we proposed a new data partitioning strategy that maximises the use of all available **labelled** images. The entire labelled pool of **4,191 images** (official Train + Validation) is used as the base for our split, through the following steps:

     (a) **Combine** the official Train and Validation labelled pools into a single pool of 4,191 images.

     (b) **Stratified 80/20 random split** of this combined pool, performed **separately** on the Urban and Rural subsets to preserve the domain balance.

     (c) **Random shuffling** of both splits for reproducibility (seed = 42).

The resulting distribution is summarised in Table 5.4.

Table 5.4: Proposed 80/20 custom split for the TransUNet model.

Subset	**Images**	**% of Labelled**	**Usage**
Train	3,352	≈80%	Model training (↑ +33% vs official)
Validation	839	≈20%	Tuning / EarlyStopping / Evaluation
Total	**4,191**	**100%**	

Table 5.5: Urban/Rural distribution in the proposed 80/20 custom split for the TransUNet model.

Subset	**Urban**	**Rural**	**Total**
Train	≈1,467	≈1,886	**3,352**
Validation	≈366	≈472	**839**
Total	≈1,833	≈2,358	**4,191**

Methodological Note: The proposed 80/20 split is constructed exclusively from the 4,191 labelled images of the official LoveDA Train+Val pool. The official Test set (1,796 images) is excluded from all quantitative experiments as it provides no ground-truth masks. Urban/Rural balance is preserved in both subsets. Since the Kaggle dataset uses a flat directory structure without district-level subfolders, strict spatial independence between splits cannot be fully guaranteed, and minor geographic overlap remains possible. Validation metrics may therefore be slightly optimistic compared to those obtained with the strict official benchmark split.

5. **Data Normalization:** Images are normalised using the `preprocess_input` function from `tf.keras.applications.resnet_v2`, which applies the ResNet50V2-specific ImageNet normalisation. This function converts RGB pixel values to the range $[-1,\ 1]$ using the formula:

$$x_{\text{norm}} = \frac{x}{127.5} - 1.0 \tag{5.1}$$

Mask labels are stored as integer tensors in the range $[0, 6]$ (after remapping from $[1, 7]$) and are **not** one-hot encoded at loading time, since the hybrid loss function accepts raw integer labels and performs the one-hot encoding internally via `tf.one_hot`.

5.4.2.2 Application on Land Cover.ai

1. **Architecture:** For this second model, we implement a hybrid TransUNet v2 architecture that combines the power of Convolutional Neural Networks (CNNs) in feature-extraction with the global context modeling capabilities of Transformers.

- **The Encoder (Backbone):** We use a pre-trained ResNet50 network on ImageNet to extract spatial features at different resolution levels (s_1, s_2, s_3, s_4). The first three feature maps used as skip connections, while the final map (s_4) is projected and flattened into sequences of patches ($32 \times 32 = 1024$patches with a dimension of 512).
- **The Bottleneck:** A block with 12 Transformer layers and a multi-head attention mechanism helps process these patches, which are enriched with a positional embedding. This setup helps the network to capture long-range contextual dependencies across the entire image.
- **The Decoder:** After the Transformer stage, the sequence outputs are reshaped back into 2D feature maps. The decoder then reconstructs the image step-by-step using bilinear upsampling, refined by 3×3 convolutions and Batch Normalization. To prevent losing fine details, we feed the high-resolution skip connections from the CNN encoder straight into this pipeline. This merges early spatial features back into the mix, allowing the model to accurately recover sharp ground structures like roads and building edges before a final Softmax layer handles the pixel-by-pixel classification.

5.10.

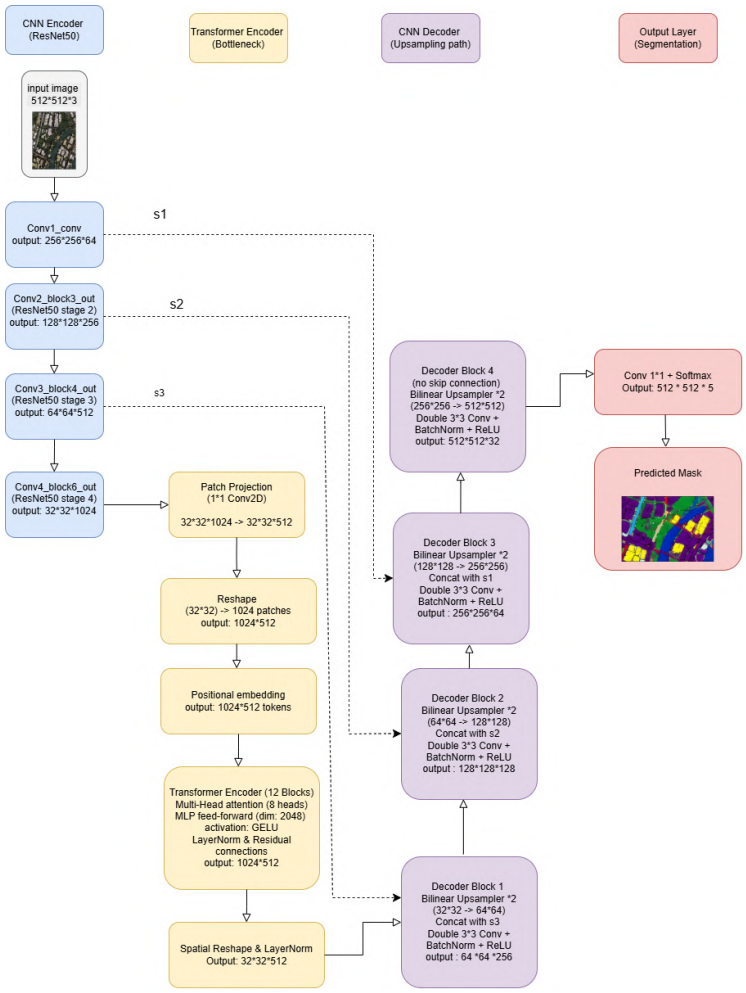


Figure 5.10: Architecture diagram of the hybrid TransUNet model applied to the LandCover.ai dataset.

2. **Data Loading:** To prevent system crashes from **Out Of Memory (OOM)** errors caused by loading thousands of high-resolution satellite images at once, we use a dynamic loading strategy via the load_and_preprocess_image function. This function reads and prepares files on the fly only when needed for a batch. It loads the aerial image (JPEG), resizes it, and normalizes the pixel values to $[0, 1]$ to speed up training. Simultaneously, it loads the label mask (PNG) using **nearest-neighbor** resizing. This specific interpolation method is critical because it keeps class IDs as clean integers, ensuring boundaries (like roads or buildings) do not blur into corrupt, non-existent categories.

```
@tf.function
def load_and_preprocess_image(img_path, mask_path):
    # 1. Read and decode the raw aerial image from disk
    img = tf.io.read_file(img_path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, [IMG_SIZE, IMG_SIZE])
    img = tf.cast(img, tf.float32) / 255.0  # Scale pixel values
   to [0, 1]
```

```
    # 2. Read and decode the label mask from disk
    mask = tf.io.read_file(mask_path)
    mask = tf.image.decode_png(mask, channels=1)

    # Nearest neighbor preserves exact class IDs without blurring edges
    mask = tf.image.resize(mask, [IMG_SIZE, IMG_SIZE], method='nearest')
    mask = tf.cast(mask, tf.int32)

    return img, mask
```

Listing 5.8: Dynamic Data Loading and Preprocessing Function

3. **Data Augmentation:** Semantic segmentation of high-resolution satellite imagery is highly sensitive to variations in viewpoint and sensor orientation. To make our hybrid model more robust and prevent overfitting, we apply random geometric transformations directly inside the TensorFlow computation graph. Each image-mask pair has a 50% chance of undergoing a horizontal flip, a vertical flip, or an orthogonal rotation (90°, 180°, or 270°). These steps maintain strict spatial alignment between the aerial scene and its target labels. Finally, the integer masks are converted into one-hot encoded vectors across the 5 target classes to match the network's output format.

```
@tf.function
def train_augment(img, mask):
    # Synchronized geometric flips on the tensor graph
    if tf.random.uniform([]) > 0.5:
        img = tf.image.flip_left_right(img)
        mask = tf.image.flip_left_right(mask)

    if tf.random.uniform([]) > 0.5:
        img = tf.image.flip_up_down(img)
        mask = tf.image.flip_up_down(mask)

    if tf.random.uniform([]) > 0.5:
        k = tf.random.uniform([], minval=1, maxval=4, dtype=tf.int32)
        img = tf.image.rot90(img, k)
        mask = tf.image.rot90(mask, k)

    # Squeeze single channel and expand to categorical depth space
    mask = tf.one_hot(tf.squeeze(mask, axis=-1), depth=NUM_CLASSES)
    return img, mask
```

Listing 5.9: TensorFlow graph implementation of synchronized geometric augmentations and categorical one-hot formatting.

4. **Data Split:** The dataset organization follows the official LandCover-AI benchmark split. The original large satellite images are divided into non-overlapping

512 × 512 pixel patches. We used the official configuration files to split these patches into training (70%), validation (15%), and testing (15%) subsets. The final distribution is summarized in Table 5.6.

Table 5.6: Distribution of LandCover-AI images into Train, Validation, and Test subsets.

Dataset Subset Split	**Image Count (Pairs)**	**Percentage Allocation (%)**
Training (`train`)	7,470	69.98%
Validation (`val`)	1,602	15.01%
Testing (`test`)	1,602	15.01%
Total Dataset	**10,674**	**100.00%**

- **Train Set:** Shuffled randomly at the start of each epoch. This keeps the data order unpredictable so the model learns real features instead of memorizing the sequence of the scenes.
- **Validation Set:** Remains completely fixed with no data augmentation. It is used at the end of each epoch to calculate the Mean IoU and decide when to save the best checkpoint (`model.keras`).
- **Test Set:** Kept separate from the training process. It is used only for final evaluation and visual predictions to check how well the model generalizes to unseen areas.

5. **Data Normalization:** To stabilize gradient descent and accelerate the convergence of the network layers, input images must be scaled to a uniform range. We implement a straightforward pixel-level normalization within the loading function by casting the raw image arrays to floating-point values and dividing them by the maximum 8-bit integer value:

```
    # Scale pixel features from [0, 255] to a stable [0, 1] range
    img = tf.cast(img, tf.float32) / 255.0

```

Listing 5.10: Data normalization implementation.

This simple adjustment scales all input features into a steady $[0, 1]$ distribution, which prevents gradients from exploding during the early stages of training.

5.5 Model Training and Hyperparameters

5.5.1 Model 1: U-Net ResNet50

5.5.1.1 Training Configuration on Dataset 1 (LoveDA):

To train our model based on the U-Net architecture with a ResNet50 encoder on the LoveDA dataset, particular attention was paid to the choice of hyperparameters, as these strongly influence the performance of the model:

- **Optimizer**: we used the **SGD optimizer with Nesterov momentum**, with an initial learning rate of 0.01, a momentum of 0.9, and a weight decay of 10^{-4}. SGD with momentum is the standard optimizer for semantic segmentation benchmarks and provides more stable convergence compared to adaptive optimizers when combined with a polynomial learning rate schedule.
- **Loss Function**: the loss function is a **composite of two complementary terms** with equal weights: **Dice Loss** and **Categorical Focal Loss** with $\gamma = 2.0$. The Dice Loss addresses class imbalance by measuring the overlap between predicted and ground-truth masks, while the Focal Loss focuses training on hard and minority-class examples by down-weighting well-classified pixels.
- **Epochs / Batch size**: we set the maximum number of epochs to **35** and the batch size to **8 images** per iteration, representing a balance between GPU memory constraints and training stability.
- **Input size**: all images are resized to **512 × 512 pixels** to ensure compatibility with the ResNet50 backbone pre-trained on ImageNet.

To improve model performance while avoiding overfitting, several regularisation and optimisation techniques were implemented:

1. **Polynomial Learning Rate Scheduler (PolyLR)**: instead of a fixed learning rate, we adopted a polynomial decay schedule with power = 0.9 and a maximum of 15,000 iterations. This schedule progressively reduces the learning rate as training advances, allowing the model to converge smoothly without oscillating around the optimum.
2. **EarlyStopping**: automatically interrupts training if the validation mIoU does not improve for **15 consecutive epochs**, while restoring the best weights obtained. This prevents the model from continuing to fit noise in the training data.
3. **Dropout**: a Dropout layer with rate 0.2 is inserted before the final output layer, randomly deactivating a portion of neurons during each training iteration to encourage the model to learn more robust representations.
4. **Weight Decay**: a weight decay of 10^{-4} is applied directly through the SGD optimizer, acting as an L2 regularisation term that penalises large weights and further reduces the risk of overfitting.

The complete training configuration is summarised in Table 5.7.

Table 5.7: Training hyperparameters of the U-Net ResNet50 model on LoveDA.

Hyperparameter	**Value**
Optimizer	SGD + Nesterov Momentum
Initial Learning Rate	0.01
Momentum	0.9
Weight Decay	10^{-4}
Loss Function	Dice (0.5) + Focal Loss (0.5, γ=2.0)
Batch Size	8
Maximum Epochs	35
LR Scheduler	Polynomial Decay (power=0.9, max_iter=15,000)
Early Stopping Patience	15 epochs
Dropout Rate	0.2
Input Image Size	512×512
Steps per Epoch	315

5.5.1.2 Training Configuration on Dataset 2 (LandCover.ai)

To train our baseline U-Net model with the ResNet50 encoder on the LandCover.ai dataset, we adjusted our hyperparameters to handle the specific characteristics of this high-resolution aerial imagery:

- **Optimizer:** We used the AdamW optimizer with an initial learning rate of 10^{-4} (0.0001) and a weight decay of 10^{-2} (0.01). This choice helps decouple the weight decay updates from the gradients, allowing the deep CNN layers to regularize properly and converge smoothly.
- **Loss Function:** Since the LandCover.ai dataset has a high class imbalance, we applied a 50/50 hybrid loss function. It combines Weighted Categorical Cross-Entropy (WCE) and Dice Loss. The cross-entropy loss includes specific class weights of $[0.8, 1.5, 1.0, 1.2, 1.5]$ to make the network focus more on smaller, harder classes like buildings and roads, while the Dice loss focuses on maximizing the pixel overlap.
- **Epochs / Batch Size:** The training process was run for 30 epochs with a batch size of 4 images per step. This specific configuration balanced our local GPU hardware constraints with stable, smooth gradient optimization steps.
- **Input Size:** All input tiles and labels were maintained at a uniform size of 512×512 pixels to remain fully compatible with the pre-trained feature dimension rules of the ResNet50 backbone.

To keep the training process efficient and prevent the model from overfitting or memorizing noise, we included three functional Keras callbacks:

1. **ReduceLROnPlateau:** This scheduler keeps track of the validation mean IoU (mIoU). If the performance plateaus for 5 consecutive epochs, it automatically drops the learning rate by 80% (multiplying by a factor of 0.2) to let the model refine its weights at a smaller step size.

2. **EarlyStopping:** To save training time and protect against overfitting, this callback monitors the validation mIoU and cuts off the training process if there is no improvement for 10 straight epochs. It also reloads the best weights recorded.

3. **ModelCheckpoint:** This tool tracks the validation mIoU at every epoch and saves the best iteration straight to disk as `best_landcover_unet.keras`.

The full list of training hyperparameters for this baseline CNN run is summarized in Table 5.8.

Table 5.8: Training hyperparameters of the U-Net ResNet50 model on LandCover.ai.

Hyperparameter	**Value**
Optimizer	AdamW
Initial Learning Rate	1×10^{-4}
Weight Decay	1×10^{-2}
Loss Function	Hybrid WCE (0.5) + Dice Loss (0.5)
Batch Size	4
Maximum Epochs	30
LR Scheduler	ReduceLROnPlateau (Factor=0.2, Patience=5)
Early Stopping	Patience 10 epochs
Dropout Rate	0.2
Input Image Size	512×512
Steps per Epoch	400

5.5.2 Model 2: TransUNet ResNet50V2

5.5.2.1 Training Configuration on Dataset 1 (LoveDA):

To train our TransUNet model with ResNet50V2 encoder on the LoveDA dataset, particular attention was paid to the choice of hyperparameters and regularisation strategies, as the hybrid CNN–Transformer architecture with 51M parameters is more prone to overfitting than pure CNN models, especially with limited training data.

- **Optimizer**: we used the **AdamW** optimizer with an initial learning rate of 10^{-4}, a weight decay of 4×10^{-4}, and gradient clipping with `clipnorm` = 1.0. AdamW decouples weight decay from the gradient update, providing more effective regularisation than standard Adam with L2 penalty, which is particularly beneficial for Transformer-based architectures.

- **Loss Function**: the loss function is a **hybrid composite loss** combining two complementary terms with equal weights:

$$\mathcal{L}_{\text{total}} = 0.5 \cdot \mathcal{L}_{\text{CE}}^{\text{smooth}} + 0.5 \cdot \mathcal{L}_{\text{Dice}}^{\text{weighted}} \tag{5.2}$$

where $\mathcal{L}_{\text{CE}}^{\text{smooth}}$ is a **weighted cross-entropy** with **label smoothing** ($\epsilon = 0.15$):

$$\tilde{y} = y \cdot (1 - \epsilon) + \frac{\epsilon}{C} \tag{5.3}$$

and $\mathcal{L}_{\text{Dice}}^{\text{weighted}}$ is a **class-weighted Dice loss** with weights:

$$\mathbf{w} = [0.5,\ 2.0,\ 2.0,\ 1.5,\ 2.0,\ 1.2,\ 1.2] \tag{5.4}$$

corresponding to Background, Building, Road, Water, Barren, Forest, and Agricultural respectively.

- **Epochs / Batch size**: we set the maximum number of epochs to **45** and the batch size to **8 images** per iteration.
- **Input size**: all images are resized to **512 × 512 pixels**, producing a sequence of $n = 32 \times 32 = 1024$ tokens at the Transformer bottleneck.
- **Dropout**: stochastic dropout increasing progressively from 0.30 to 0.40 over the 12 Transformer blocks, plus `SpatialDropout2D` with rate 0.40 after each decoder convolutional block.

To improve model performance while avoiding overfitting, the following techniques were implemented:

1. **Warmup + Cosine Decay Scheduler**: a custom two-phase learning rate schedule:
   - **Linear warmup phase** (first 5 epochs): the learning rate increases linearly from 0 to $\eta_0 = 10^{-4}$.

$$\eta(t) = \eta_0 \cdot \frac{t}{t_{\text{warmup}}} \quad \text{for } t < t_{\text{warmup}} \tag{5.5}$$

   - **Cosine decay phase** (remaining epochs): the learning rate decays smoothly down to $\alpha = 10^{-7}$:

$$\eta(t) = \alpha + \frac{1}{2}(\eta_0 - \alpha)\left(1 + \cos\left(\pi \cdot \frac{t - t_{\text{warmup}}}{t_{\text{total}} - t_{\text{warmup}}}\right)\right) \tag{5.6}$$

2. **EarlyStopping**: training is automatically interrupted if the validation mIoU does not improve for **12 consecutive epochs**, while restoring the best model weights.
3. **Gradient Clipping**: a maximum gradient norm of 1.0 is enforced via `clipnorm` in the AdamW optimizer, preventing gradient explosion during the early training phases.
4. **Weight Decay**: a weight decay of 4×10^{-4} is applied through the AdamW optimizer, acting as a decoupled L2 regularisation across all trainable layers.

The complete training configuration is summarised in Table 5.9.

Table 5.9: Training hyperparameters of the TransUNet ResNet50V2 model on LoveDA.

Hyperparameter	**Value**
Optimizer	AdamW
Initial Learning Rate	10^{-4}
Weight Decay	4×10^{-4}
Gradient Clipping	`clipnorm` = 1.0
Loss Function	Weighted CE (0.5) + Weighted Dice (0.5)
Label Smoothing	$\epsilon = 0.15$
Class Weights	$[0.5, 2.0, 2.0, 1.5, 2.0, 1.2, 1.2]$
Batch Size	8
Maximum Epochs	45
LR Scheduler	Warmup (5 ep.) + Cosine Decay
Early Stopping Patience	12 epochs
Dropout Transformer	0.30
Dropout Decoder	`SpatialDropout2D` = 0.40
Transformer Blocks	12
Attention Heads	8
Token Dimension	512
FFN Hidden Dimension	2,048
Input Image Size	512×512
Steps per Epoch	315 (official) / 419 (80/20)
Total Parameters	51,327,399
Trainable Parameters	51,300,007 (99.9%)

5.5.2.2 Training Configuration on Dataset 2 (LandCover.ai):

To train our hybrid TransUNet model on the LandCover.ai dataset, we selected our training hyperparameters to carefully balance the initialization of the pre-trained ResNet50 CNN layers while preventing gradient explosion within the Transformer attention blocks:

- **Optimizer:** we used the AdamW optimizer with an initial learning rate of 10^{-4}, a weight decay of 1×10^{-2}, and gradient clipping via a maximum gradient norm threshold of `clipnorm = 1.0`. AdamW decouples weight decay from the gradient update step, providing more effective regularization across the high-parameter network than standard Adam with an L_2 penalty.

- **Loss Function:** the loss function is a custom hybrid composite loss (`hybrid_loss`) combining two complementary optimization terms with equal weight configurations:

$$\mathcal{L}_{\text{total}} = 0.5 \cdot \mathcal{L}_{\text{Focal}} + 0.5 \cdot \mathcal{L}_{\text{Dice}} \tag{5.7}$$

  where $\mathcal{L}_{\text{Focal}}$ is a Categorical Focal Loss ($\gamma = 2.0$) computed using target label

smoothing ($\alpha = 0.1$) applied directly onto the ground-truth arrays:

$$y_i^{\text{smooth}} = y_i \cdot (1 - \alpha) + \frac{\alpha}{C} \tag{5.8}$$

for $C = 5$ land-cover classes. The term $\mathcal{L}_{\text{Dice}}$ represents a Soft Dice Loss designed to optimize boundary-level intersections across all categories simultaneously.

- **Epochs / Batch size:** we set the maximum number of training epochs to 24 and the global batch size to 8 images per iteration. This data allocation is synchronized across two active devices (a localized batch size of 4 images per active GPU) managed via TensorFlow's distributed `MirroredStrategy` scope.
- **Input size:** aerial input image segments and masks are restricted to a uniform size of 512×512×3 pixels. Feature vectors extracted from the `conv4_block6_out` layer of the CNN backbone are projected into a token dimension ($d_{\text{model}} = 512$), creating a sequence length of $n = 32 \times 32 = 1,024$ patches at the Transformer bottleneck layers.
- **Dropout:** to strictly prevent co-adaptation of neurons, a progressive dropout scaling strategy is integrated across the 12 sequential Transformer blocks, rising linearly from a base rate of 0.10 up to a structural ceiling of 0.30.

To optimize model convergence while guarding against hardware resource limits and overfitting, the following computational techniques were implemented within the runtime script:

1. **Warmup + Cosine Decay Scheduler:** a custom `WarmupCosineDecay` learning rate schedule was implemented to handle step transitions based on total epoch steps (T_{total}):
   - *Linear warmup phase (first 5 epochs):* the operational step learning rate increases linearly from an initial near-zero baseline up to the peak target velocity of $\eta_{\max} = 10^{-4}$:

$$Lr(t) = \eta_{\max} \cdot \frac{t}{T_{\text{warmup}}} \quad \text{for } t < T_{\text{warmup}} \tag{5.9}$$

   - *Cosine decay phase (remaining 19 epochs):* the learning rate updates scale smoothly down toward an absolute fine-tuning baseline floor of $\eta_{\min} = 10^{-7}$:

$$Lr(t) = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left[1 + \cos\left(\pi \cdot \frac{t - T_{\text{warmup}}}{T_{\text{total}} - T_{\text{warmup}}}\right)\right] \tag{5.10}$$

2. **EarlyStopping:** tracking validation performance at the end of each epoch, the training run automatically interrupts execution if the macro validation Mean IoU (`val_iou`) fails to improve or set a new record for 12 consecutive epochs, while cleanly restoring the optimal recorded weights.

3. **ModelCheckpointing:** monitors validation scores sequentially and preserves the highest-performing network state to disk as an immutable checkpoint file named `best_transunet_v2.keras`.

4. **CSVLogger:** automatically maintains an external trace file (`training_log_transunet_v2.csv`) recording comprehensive loss values, accuracy charts, and metric fluctuations across the operational training lifetime.

The full operational training and structural design hyperparameter configurations are systematically summarized in Table 5.10.

Table 5.10: Training hyperparameters of the TransUNet Hybrid model on LandCover.ai.

Hyperparameter	**Value**
Distributed Infrastructure	TensorFlow MirroredStrategy (Dual-GPU Scope)
Global Batch Size (B_{global})	8 patches (4 images per localized device)
Optimizer	AdamW
Initial Peak Learning Rate ($\eta_{\max}$)	1×10^{-4}
Weight Decay Coefficient (λ)	1×10^{-2}
Gradient Clipping Limit	1.0 (`clipnorm`)
Learning Rate Scheduler	Warmup (5 epochs) + Cosine Decay
Baseline Learning Rate Floor ($\eta_{\min}$)	1×10^{-7}
Loss Formulation	Composite Loss: $0.5 \times$ Focal ($\gamma = 2.0$) + $0.5 \times$ Dice
Target Label Smoothing (α)	0.1
Transformer Dropout Rate	Progressive step tracking (0.10 to 0.30 Max)
Input Spatial Image Size	$512 \times 512 \times 3$ pixels
Maximum Horizon Epochs	24 epochs
Early Stopping Patience	12 epochs monitored on Validation mIoU
Total Class Categories (C)	5 (Background, Building, Woodland, Water, Road)
Transformer Blocks (N)	12
Attention Heads (n_{heads})	8
Token Bottleneck Dimension (d_{model})	512
FFN Hidden Dimension (`mlp_dim`)	2,048

5.6 Results and Discussion

5.6.1 Dataset 1: LoveDA

The evaluation of all models was carried out on the respective validation sets. For the TransUNet model, two experiments are reported: one using the official LoveDA split and one using the proposed 80/20 custom split.

5.6.1.1 Training Phase

1. **Model 1 U-Net ResNet50**

Training stopped at **epoch 33** out of 35, with a training mIoU of **52.74%** and a validation mIoU of **50.69%**, corresponding to a composite loss of 0.2200 (train) and 0.2316 (val). Training lasted approximately **4 hours 30 minutes** on 2× Tesla T4 GPUs. The training curves are shown in Figure 5.11.

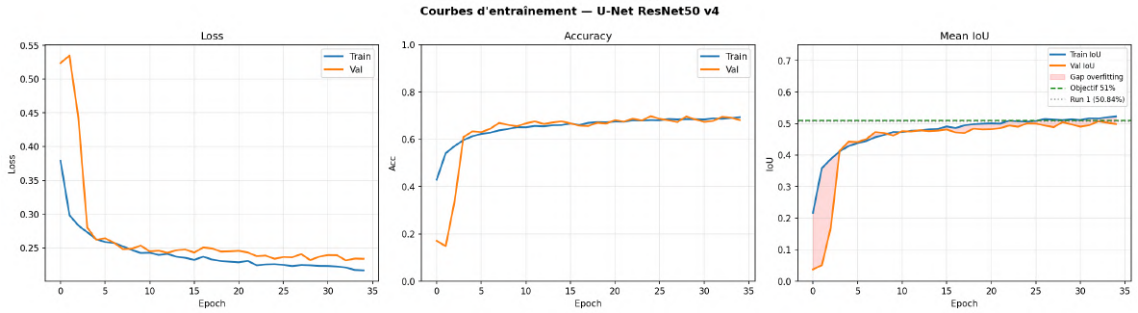


Figure 5.11: Training performance curves of the U-Net ResNet50 model on LoveDA.

2. **Model 2 TransUNet ResNet50V2 (Official Split)**

   Using the official LoveDA split (2,522 train / 1,669 val), training stopped at **epoch 44** out of 45 via EarlyStopping (patience = 12), with the best model saved at **epoch 32**. The best validation mIoU reached **44.50%**, with a training mIoU of 52.94% at that epoch, corresponding to a composite loss of 0.9276 (train) and 0.9967 (val). Training lasted approximately **11 hours** on 2× Tesla T4 GPUs.

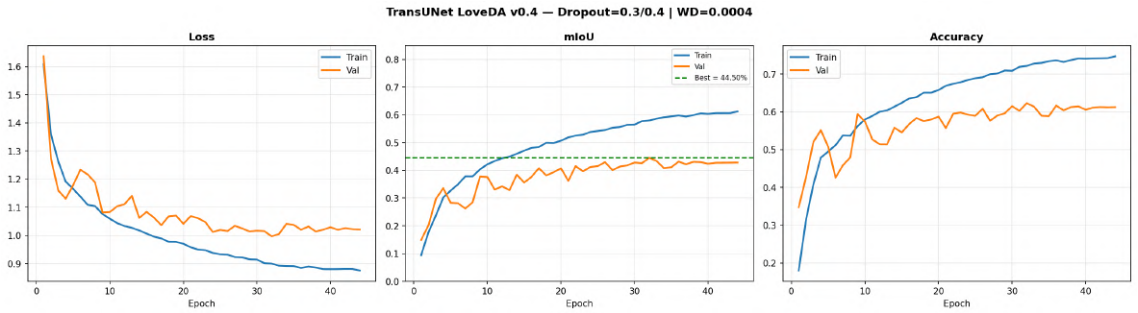


Figure 5.12: Training performance curves of the TransUNet ResNet50V2 model on LoveDA (official split).

3. **Model 2 TransUNet ResNet50V2 (Proposed 80/20 Split)**

   Using the proposed 80/20 custom split (3,352 train / 839 val), all **35 epochs** were completed without early stopping being triggered, with the best model saved at **epoch 34**. The best validation mIoU reached **55.21%**, with a training mIoU of **59.60%**, corresponding to a composite loss of **0.8692** (train) and **0.8810** (val). Training lasted approximately **10 hours 40 minutes**. The validation mIoU increased steadily across all 35 epochs without plateau, with the training–validation gap remaining controlled at 4.39%, indicating good generalisation. The fact that the model was still improving at epoch 35 suggests that additional training epochs would likely yield further gains.

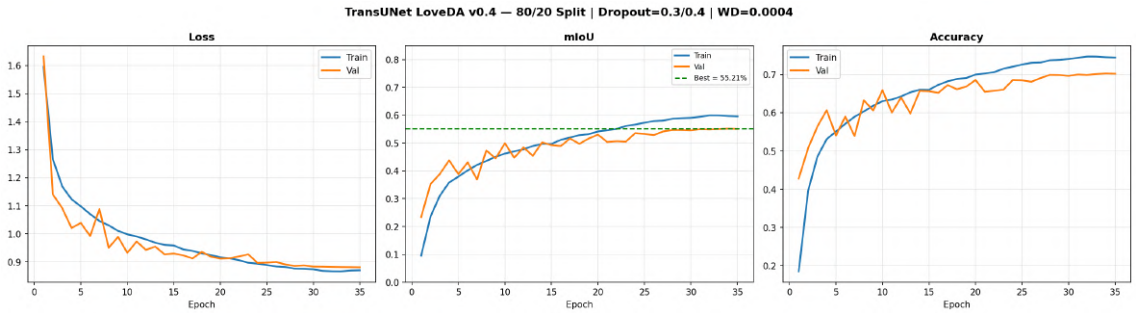


Figure 5.13: Training performance curves of the TransUNet `ResNet50V2` model on LoveDA (proposed 80/20 split).

5.6.1.2 Evaluation Metrics

The following standard semantic segmentation metrics are used:

Intersection over Union (IoU) per class:

$$\text{IoU}_c = \frac{TP_c}{TP_c + FP_c + FN_c} \tag{5.11}$$

Mean IoU (mIoU):

$$\text{mIoU} = \frac{1}{C} \sum_{c=1}^{C} \text{IoU}_c \tag{5.12}$$

F1-Score per class:

$$F1_c = 2 \times \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{5.13}$$

Dice coefficient per class (used for TransUNet):

$$\text{Dice}_c = \frac{2 \cdot TP_c}{2 \cdot TP_c + FP_c + FN_c} \tag{5.14}$$

where TP_c, FP_c, and FN_c denote true positives, false positives, and false negatives for class c respectively.

5.6.1.3 Confusion Matrix

1. **Model 1 U-Net ResNet50: Confusion Matrix Analysis**

   To analyze the pixel-level classification behavior and identify common misclassification trends between terrain classes, the normalized confusion matrix was computed. Figure 5.14 details the quantitative distribution of these predictions. The diagonal entries represent the ratio of correctly classified pixels for each individual category.

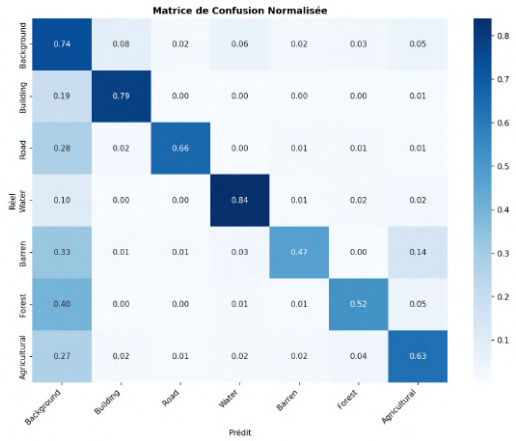
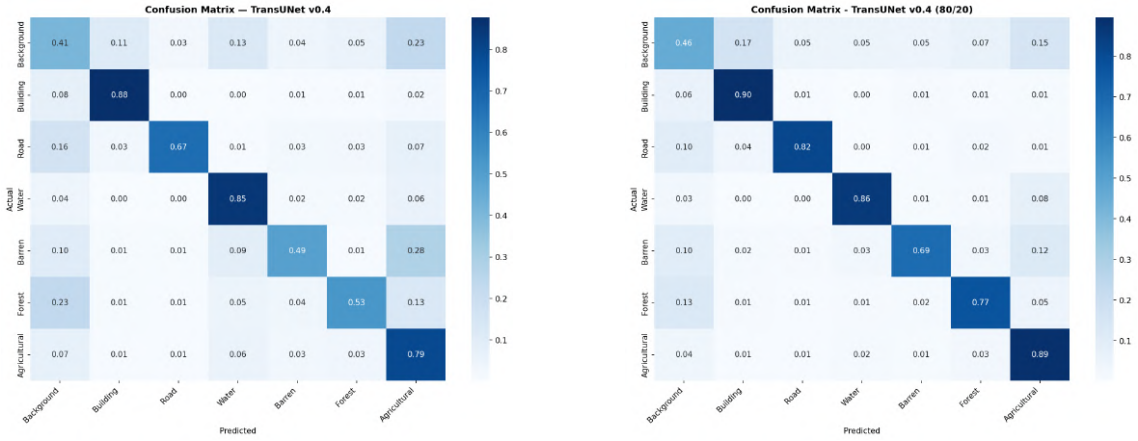


Figure 5.14: Normalized confusion matrix of the U-Net `ResNet50` model evaluated on the LoveDA validation set.

2. **Model 2 TransUNet Confusion Matrix Analysis**

   To understand the underlying pixel-misclassification trends between the dataset distribution strategies, the normalized confusion matrices for both configurations were evaluated. Figure 5.15 contrasts the class-wise prediction distributions of the TransUNet architecture. Comparing the configurations side-by-side highlights that diagonal values (correct classifications) show a significant per-class recognition improvement when utilizing the proposed 80/20 split strategy.



(a) Official Dataset Split (b) Proposed 80/20 Split

Figure 5.15: Normalized confusion matrices of the TransUNet `ResNet50V2` model evaluated on the LoveDA validation sets.

5.6.1.4 Per-Class Results

1. **Model 1 U-Net ResNet50**

   To evaluate the class-specific performance of the convolutional baseline architecture on the LoveDA dataset, the metric profiles for individual terrain

categories were extracted. Figures 5.16 and 5.17 present the class-wise breakdown for the IoU and F1-Score metrics respectively.

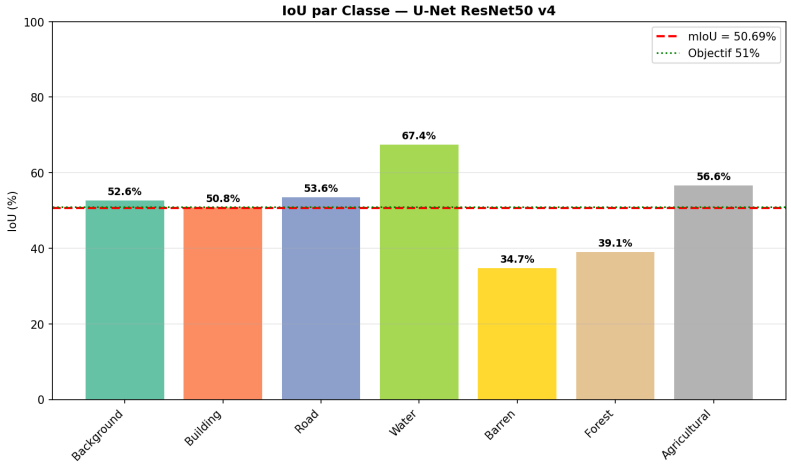


Figure 5.16: Per-class IoU (%) of the U-Net ResNet50 model on the LoveDA validation set.

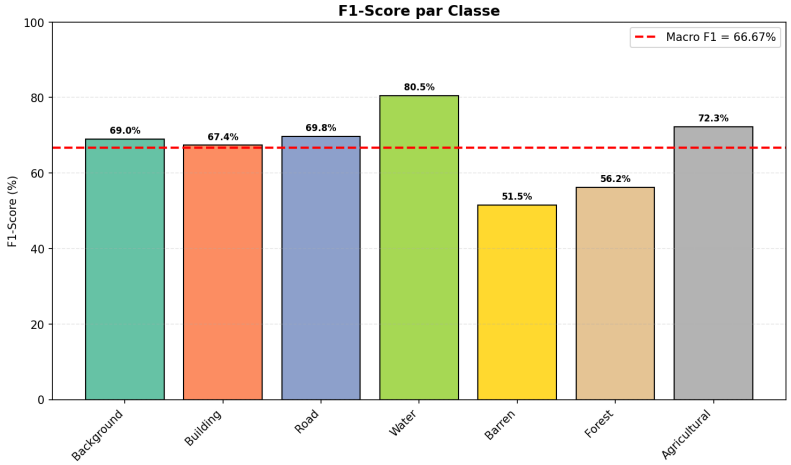


Figure 5.17: Per-class F1-Score (%) of the U-Net ResNet50 model on the LoveDA validation set.

Table 5.11: Per-class IoU and F1-Score of the U-Net ResNet50 model on the LoveDA validation set.

Class	IoU (%)	F1-Score (%)
Background	52.62	68.95
Building	50.81	67.38
Road	53.57	69.76
Water	**67.42**	**80.54**
Barren	34.71	51.53
Forest	39.07	56.19
Agricultural	56.63	72.31
mIoU / Macro F1	**50.69**	**66.67**

Key observations for U-Net ResNet50:

- The **Water** class achieves the best performance (IoU = 67.42%, F1 = 80.54%) thanks to its distinctive visual characteristics (uniform colour, homogeneous texture).
- The **Agricultural** class also achieves strong results (IoU = 56.63%) thanks to its regular textures and characteristic geometric patterns.
- The **Barren** (IoU = 34.71%) and **Forest** (IoU = 39.07%) classes are the least well classified, due to visual similarity with other classes and significant intra-class variability.

2. **Model 2 TransUNet ResNet50V2 (Official Split)**

Figures 5.18 and 5.19 present the per-class breakdown for the TransUNet model trained on the official LoveDA split.

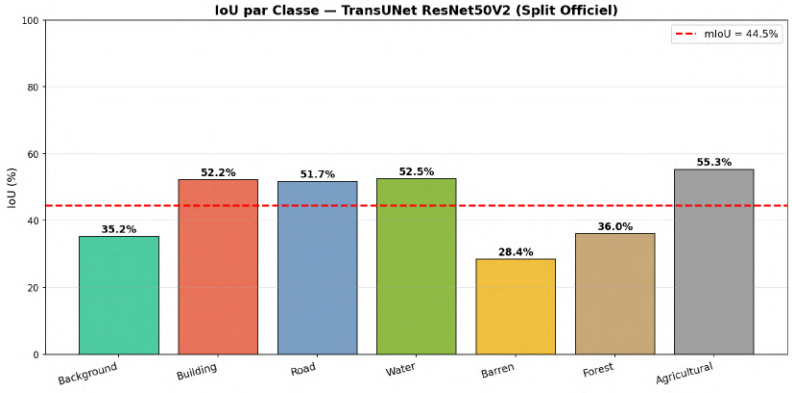


Figure 5.18: Per-class IoU (%) of the TransUNet ResNet50V2 model on the LoveDA official validation set.

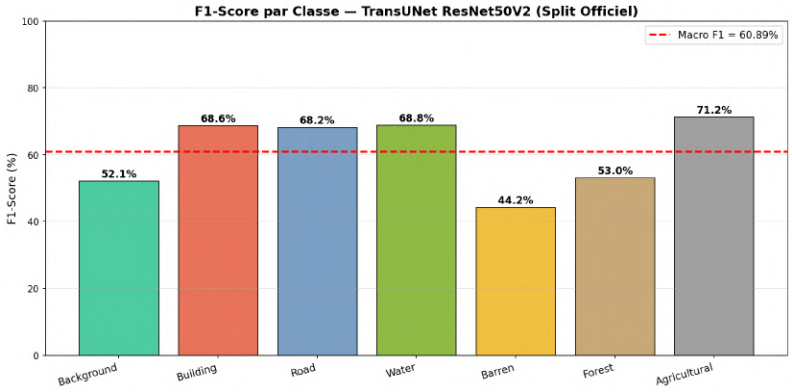


Figure 5.19: Per-class F1-Score (%) of the TransUNet ResNet50V2 model on the LoveDA official validation set.

Table 5.12: Per-class IoU, Precision, Recall, F1-Score, and Dice of the TransUNet ResNet50V2 model on the LoveDA official validation set.

Class	**IoU (%)**	**Prec (%)**	**Recall (%)**	**F1 (%)**	**Dice (%)**
Background	35.25	72.24	40.77	52.12	52.12
Building	52.25	56.36	87.75	68.63	68.64
Road	51.73	69.25	67.16	68.19	68.19
Water	52.46	57.71	85.22	68.82	68.82
Barren	28.41	40.10	49.37	44.25	44.25
Forest	36.05	53.05	52.94	53.00	53.00
Agricultural	**55.34**	**64.69**	79.28	71.25	71.25
Macro	**44.50**	**59.06**	**66.07**	**60.89**	**60.89**

3. **Model 2 TransUNet ResNet50V2 (Proposed 80/20 Split)**

   Figures 5.20 and 5.21 present the per-class breakdown for the TransUNet model trained on the proposed 80/20 split.

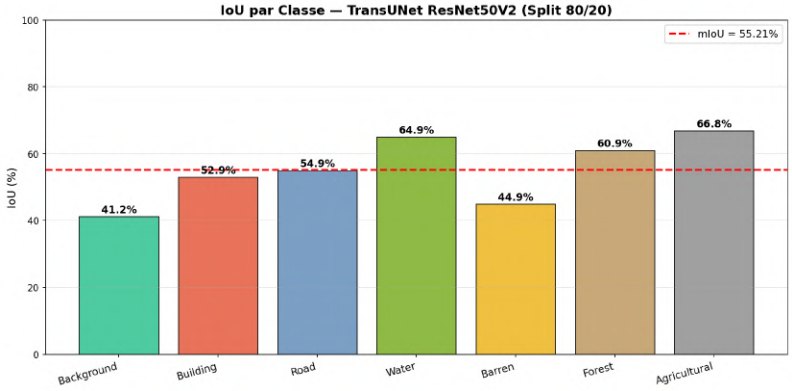


Figure 5.20: Per-class IoU (%) of the TransUNet ResNet50V2 model on the proposed 80/20 validation set.

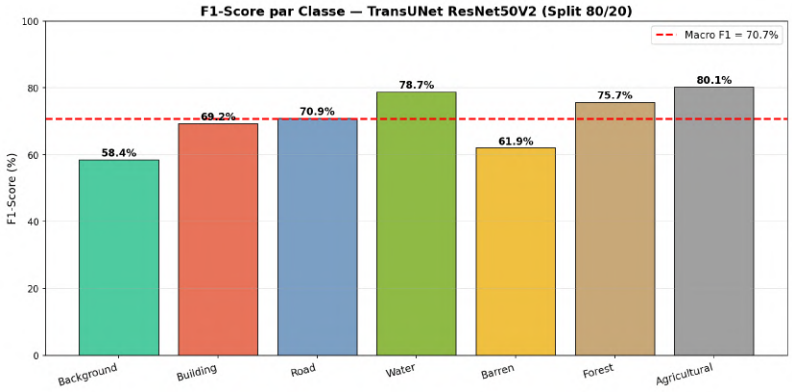


Figure 5.21: Per-class F1-Score (%) of the TransUNet ResNet50V2 model on the proposed 80/20 validation set.

Table 5.13: Per-class IoU, Precision, Recall, F1-Score, and Dice of the TransUNet ResNet50V2 model on the proposed 80/20 validation set.

Class	**IoU (%)**	**Prec (%)**	**Recall (%)**	**F1 (%)**	**Dice (%)**
Background	41.20	79.47	46.10	58.35	58.35
Building	52.92	56.38	89.61	69.22	69.22
Road	54.87	62.61	81.61	70.86	70.86
Water	64.91	72.72	85.82	78.72	78.72
Barren	44.85	56.37	68.70	61.93	61.93
Forest	60.88	74.32	77.10	75.68	75.68
Agricultural	**66.85**	72.73	**89.20**	**80.13**	**80.13**
Macro	**55.21**	**67.80**	**76.88**	**70.70**	**70.70**

Table 5.14 presents a direct per-class comparison between the two TransUNet split experiments.

Table 5.14: Per-class IoU (%) comparison

Class	**Official Split (%)**	**80/20 Split (%)**	**Δ (%)**
Background	35.25	41.20	+5.95
Building	52.25	52.92	+0.67
Road	51.73	54.87	+3.14
Water	52.46	64.91	+12.45
Barren	28.41	44.85	+16.44
Forest	36.05	60.88	+24.83
Agricultural	55.34	66.85	+11.51
mIoU	**44.50**	**55.21**	**+10.71**

This table indicates comparison between the official split and the proposed 80/20 split for the TransUNet ResNet50V2 model. Δ indicates the absolute improvement of the 80/20 split over the official split.

5.6.1.5 Visualization of Predictions

1. **Model 1 U-Net ResNet50**

   Figure 5.22 and 5.23 illustrates the qualitative performance of the convolutional baseline architecture.

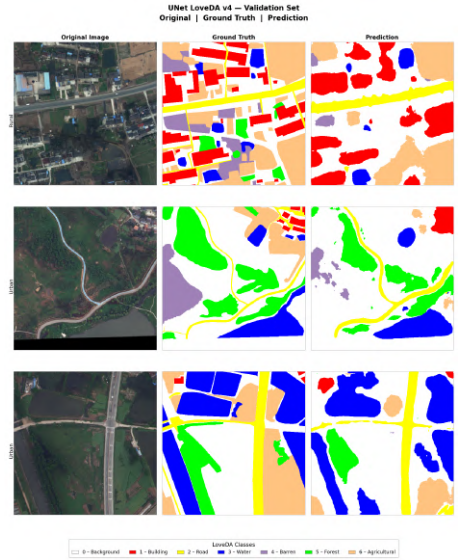


Figure 5.22: Examples of U-Net ResNet50 predictions on the LoveDA validation set.

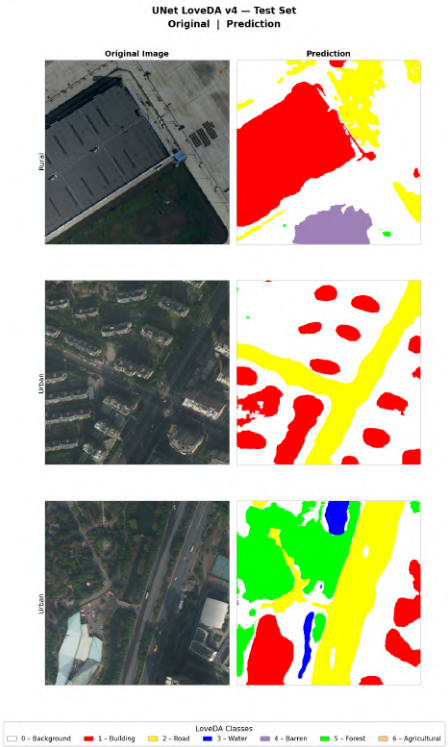


Figure 5.23: Qualitative segmentation predictions of the U-Net ResNet50 model on the LoveDA Test set (no ground-truth masks available).

2. **Model 2 TransUNet: Official Split vs. Proposed 80/20 Split**

   A qualitative evaluation was conducted to visually assess the spatial improvements gained by adjusting the dataset distribution strategy. Figure 5.24 highlights the stark comparative differences in prediction boundaries between the two training configurations. Each row represents a target sample (Rural and Urban scenes respectively). From left to right: original RGB satellite image, ground-truth segmentation mask, prediction using the official dataset split (mIoU = 44.50%), and prediction using the proposed random 80/20 split (mIoU = 55.21%). The 80/20 split demonstrates visibly superior class delineation, specifically reducing noise within the Forest, Water, and Agriculture classes.

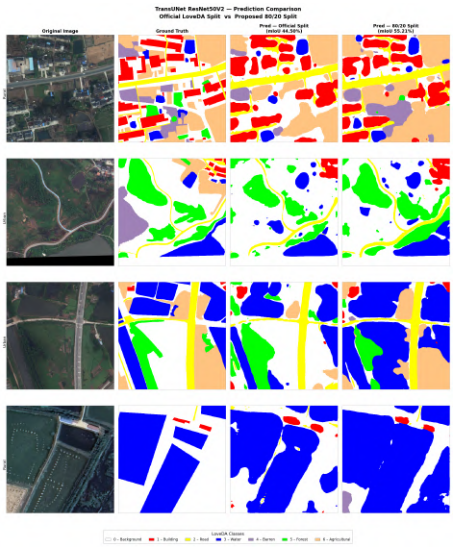


Figure 5.24: Comparison of TransUNet ResNet50V2 predictions on the LoveDA validation set across distinct terrain types .

5.6.1.6 Comparison between Model 1 and Model 2

Table 5.15 summarises the performance of all models and configurations on the LoveDA dataset.

Table 5.15: Summary comparison of all models evaluated on the LoveDA dataset.

Model	Split	mIoU (%)	F1 (%)	Params	Best Epoch	Train Time
U-Net ResNet50	Official†	50.69	66.67	32.6M	33/35	≈4h30
TransUNet ResNet50V2	Official†	44.50	60.89	51.3M	32/45	≈11h
TransUNet ResNet50V2	**80/20‡**	**55.21**	**70.70**	**51.3M**	**34/35**	**≈10h40**

The comparative analysis reveals the following key findings:

- **TransUNet 80/20 achieves the best overall performance** with mIoU = **55.21%** and F1 = **70.70%**, surpassing both U-Net ResNet50 (50.69%,

+4.52%) and TransUNet with the official split (44.50%, +10.71%). This confirms that the hybrid CNN–Transformer architecture outperforms the pure CNN approach when provided with sufficient training data.

- **Data quantity is the critical factor** for TransUNet performance. The increase from 2,522 to 3,352 training images (+33%) produced an improvement of +10.71% mIoU, while the architecture and all hyperparameters remained identical.
- **U-Net ResNet50 remains an efficient baseline** with fewer parameters (32.6M vs 51.3M) and significantly shorter training time (≈4h vs ≈8–11h), well-suited to resource-constrained platforms.
- **The TransUNet 80/20 training gap** between training mIoU (59.60%) and validation mIoU (55.21%) is moderate (4.39%), indicating good generalisation. The model was still improving at epoch 35, suggesting further gains are achievable with extended training.

5.6.1.7 Comparison with the Literature

Table 5.16 presents a comparison of our models with state-of-the-art methods on LoveDA.

Table 5.16: Comparison of our models with related works on the LoveDA dataset.

Method	Params	mIoU (%)	Split
DeepLabV3+ (cnn) [63]	—	47.62	Official
UNetResNet50 (cnn) [63]	—	47.84	Official
HRNet (cnn) [63]	—	49.79	Official
TransUNet (hybride) [123]	—	48.90	Official
UNetFormer (hybride) [123]	—	52.40	Official
Our U-Net ResNet50	**32.6M**	**50.69**	Official
Our TransUNet ResNet50V2	**51.3M**	**44.50**	Official
Our TransUNet ResNet50V2	**51.3M**	**55.21**	80/20

Our **TransUNet ResNet50V2 with the 80/20 split** achieves **55.21% mIoU**, surpassing all methods including TransUNet (paper) (48.90%, +6.31%) , with **51.3M parameters** , significantly fewer than TransUNet (paper), using only the freely available Kaggle platform (Tesla T4 GPUs).

Our **U-Net ResNet50** achieves 50.69% mIoU on the official split, surpassing all classical CNN baselines (DeepLabV3+: 47.62%) and the original TransUNet paper (48.90%) with only 32.6M parameters and ≈4 hours of training, confirming the effectiveness of the composite Dice+Focal loss and multi-scale augmentation strategy.

5.6.1.8 Difficulties Encountered

During the development of our semantic segmentation system, we encountered various complications mainly related to hardware limitations. The Google Colab plat-

form provides insufficient RAM and GPU resource limitations, which caused frequent interruptions with our 512 × 512 images and a batch size of 8. Faced with these obstacles, we migrated to the Kaggle platform, which provided higher RAM and 30h GPU access per week with 2× Tesla T4 GPUs.

Despite these improvements, GPU memory saturation when loading the full dataset simultaneously led us to implement a dynamic generator (`LoveDAGenerator`) for efficient batch-by-batch memory loading. Additionally, the XLA compilation of TensorFlow operations during the first training epoch caused significant warm-up delays, which are normal behaviour for the CUDA/XLA stack and do not affect the final model performance.

5.6.2 Dataset 2: Land Cover

We evaluated both models on the official validation split of the LandCover-AI dataset, which contains 1,602 pre-split image pieces of 512 × 512 pixels.

5.6.2.1 Training Phase

1. **Model 1 U-Net ResNet50**

   Training completed all 30 epochs smoothly without early termination. The automated check restored the optimal weights from the best performing step at **epoch 28**. At this peak checkpoint, the model reached a validation mIoU of **84.12%** and a training mIoU of **70.72%**, with an epoch training loss of 0.1910 and a validation loss of 0.1634. Training lasted approximately **6 hours 55 minutes** (averaging 830 seconds per epoch) on 2× Tesla T4 GPUs. The training curves are shown in 5.25.

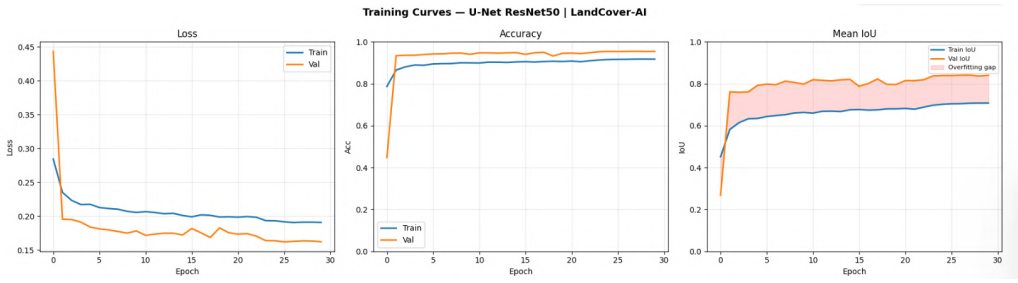


Figure 5.25: Training performance curves of the U-Net ResNet50 model on LandCover-AI.

2. **Model 2 Hybrid CNN–Transformer**

   Using the designated LandCover-AI split (7,470 train / 1,602 val), training completed all 24 epochs within a multi-GPU distributed `MirroredStrategy`. The automated tracking restored model weights from the end of the best performing step at **epoch 20**. At this peak checkpoint, the model reached a validation mIoU of **83.52%** and a Macro F1-Score of **90.58%**, corresponding to a composite validation loss of **0.9778** and a training accuracy of **95.44%**. Training lasted approximately **11 hours 25 minutes** (averaging 1,712 seconds

per epoch) on 2× Tesla T4 GPUs due to the heavy computational overhead of the self-attention mechanism layers.

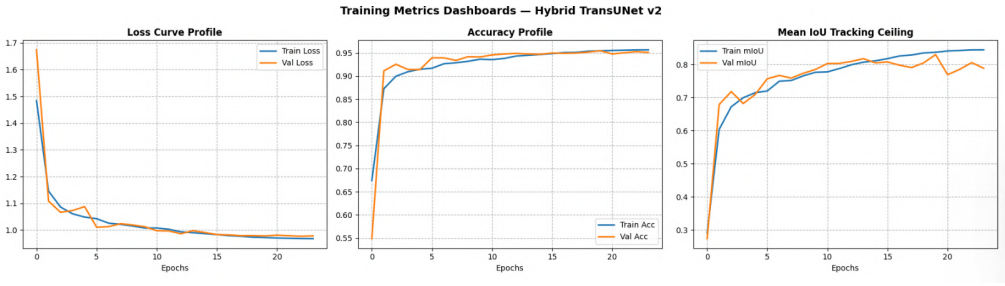


Figure 5.26: Training performance curves of the Hybrid CNN–Transformer model on LandCover-AI.

5.6.2.2 Evaluation Metrics

The following standard semantic segmentation metrics are used:

- Intersection over Union (IoU) per class:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c} \tag{5.15}$$

- Mean IoU (mIoU):

$$mIoU = \frac{1}{C} \sum_{c=1}^{C} IoU_c \tag{5.16}$$

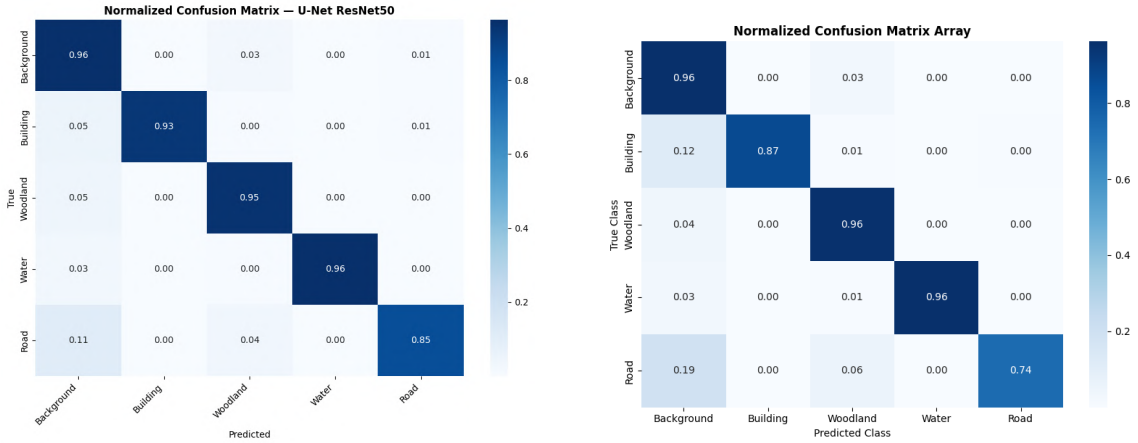
- F1-Score per class:

$$F1_c = \frac{2 \times Precision_c \times Recall_c}{Precision_c + Recall_c} = \frac{2 \cdot TP_c}{2 \cdot TP_c + FP_c + FN_c} \tag{5.17}$$

where TP_c, FP_c, and FN_c denote true positives, false positives, and false negatives for class c respectively.

5.6.2.3 Confusion Matrix

To evaluate how well our models predict each category, we analyzed their normalized confusion matrices. Figure 5.27 shows the performance of both the baseline U-Net and our proposed Hybrid architecture side-by-side. The diagonal values represent the percentage of correctly classified pixels for each class.



(a) Baseline U-Net ResNet50 (b) Proposed Hybrid CNN–Transformer

Figure 5.27: Normalized confusion matrices evaluated on the LandCover-AI validation set.

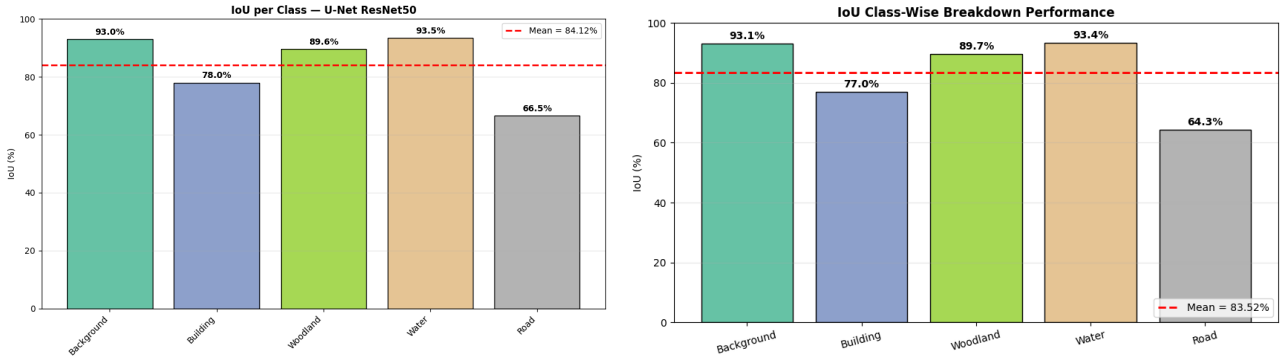
Analysis and Key Discussion:

The diagonal values show that the Hybrid model (Figure 5.27b) generally achieves higher scores, meaning it classifies pixels more accurately than the standard U-Net. This improvement is very clear for complex structures like **Buildings** and **Roads**. While the traditional U-Net often confuses narrow roads with background soil because of its limited field of view, our Hybrid model fixes many of these mistakes. By using Transformer layers, the network understands the global layout of the image, helping it follow continuous roads without breaking their edges.

However, some small errors remain for both models. Specifically, there is still some confusion between **Woodland** (trees) and **Background** (fields/grass). This happens because these classes look very similar in color and texture on satellite images. Overall, the cleaner diagonal lines of the Hybrid model prove that combining local convolutions with global attention blocks helps the system better understand both fine details and large structures.

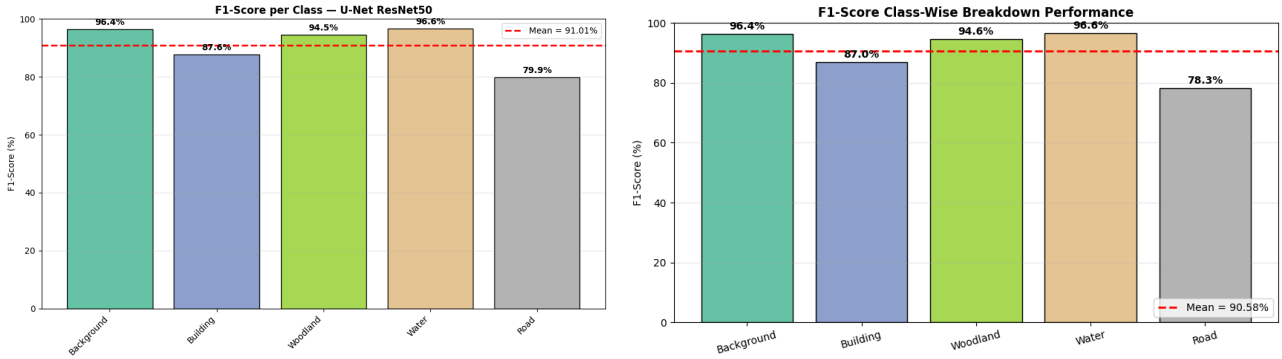
5.6.2.4 Per-Class Results

To better analyze where each model excels, the class-wise breakdown for both IoU and F1-Score metrics was visualized. Figure 5.28 and Figure 5.29 illustrate the individual performance profiles for the U-Net baseline and the proposed Hybrid network side-by-side.



(a) Model 1 (U-Net ResNet50) (b) Model 2 (Hybrid Transformer)

Figure 5.28: Class-wise Intersection over Union (IoU %) breakdown evaluated on the LandCover-AI validation set for both architectures.



(a) Model 1 (U-Net ResNet50) (b) Model 2 (Hybrid Transformer)

Figure 5.29: Class-wise F1-Score (%) breakdown evaluated on the LandCover-AI validation set for both architectures.

This table 5.17 compares (IoU and F1-Score) on the LandCover-AI validation set between the baseline U-Net (Epoch 28) and the proposed Hybrid CNN-Transformer (Epoch 24). Δ indicates the absolute performance difference.

Table 5.17: Comprehensive per-class performance comparaison

	IoU (%)			**F1-Score (%)**		
Class	**U-Net**	**Hybrid**	Δ	**U-Net**	**Hybrid**	Δ
Background	92.99	93.07	+0.08	96.37	96.41	+0.04
Building	78.01	77.03	-0.98	87.65	87.02	-0.63
Woodland	89.58	89.75	+0.17	94.50	94.60	+0.10
Water	93.48	93.44	-0.04	96.63	96.61	-0.02
Road	66.53	64.30	-2.23	79.90	78.27	-1.63
Average / Macro	**84.12**	**83.52**	**-0.60**	**91.01**	**90.58**	**-0.43**

Key Observations of Model 1 (U-Net ResNet50):

- The **Water** class achieves the absolute highest score ($IoU = 93.48\%$, $F1 = 96.63\%$) because of its stark, uniform color distribution and highly distinct geographical borders.
- The **Road** class remains the lowest score ($IoU = 66.53\%$), reflecting the inherent difficulty of parsing narrow linear infrastructure elements. However, the ResNet50 encoder shows strong localized texture discrimination across the background classes.

Key Observations of Model 2 (Hybrid CNN–Transformer):

- Performance remains balanced across targets, staying well above 93% IoU for both **Water** and **Background** landscapes.
- The transformer layers provide stable global context tracking, achieving highly competitive values across structurally complex geometries.

Discussion of Results:

The comparative summary in Table 5.17 yields an insightful architectural trade-off. While Model 2 shows minor improvements in broad continuous textures such as **Background** (+0.08%) and **Woodland** (+0.17%), the convolutional Model 1 retains a slight margin on edge-heavy, narrow objects like **Roads** (+2.23%) and **Buildings** (+0.98%).

This indicates that for high-contrast, high-resolution imagery found in LandCover-AI, the intensive local pixel localization of traditional convolutional encoders remains highly efficient. The hybrid model provides high stability across wide spaces but faces a minor smoothing effect near very narrow infrastructure limits due to tokenization resolutions.

5.6.2.5 Visualization of Predictions

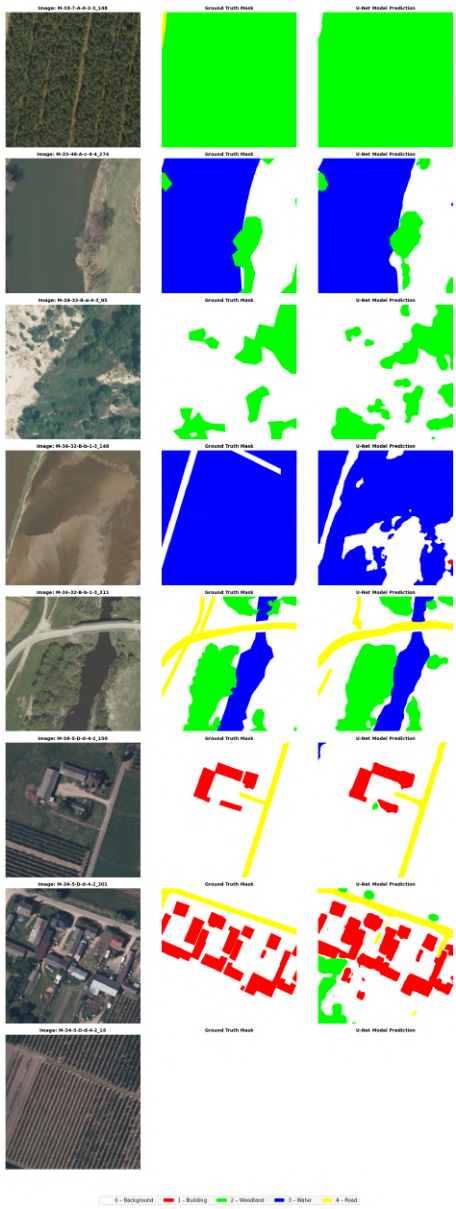


Figure 5.30: Examples of U-Net ResNet50 predictions on the LandCover-AI validation set.

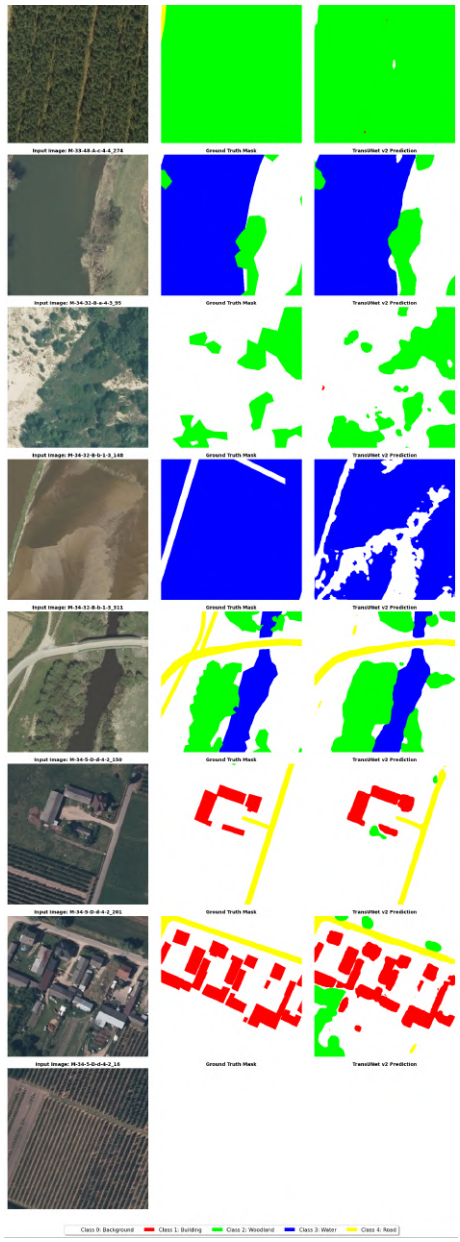


Figure 5.31: Examples of Hybrid CNN–Transformer predictions on the LandCover-AI validation set.

5.6.2.6 Comparison with the Literature

Table 5.18 presents a comparison of our models with state-of-the-art CNN-based methods on the LandCover-AI dataset. Since no hybrid CNN–Transformer architecture has been previously reported for this dataset, we focus the comparison on F1-Score, the metric consistently reported in the literature.

Table 5.18: Comparison of our models with related CNN-based works on the LandCover-AI dataset.

Method	Backbone	Params	Background	Building	Woodland	Water	Road	Macro F1 (%)
U-Net [124]	—	—	0.92	0.76	0.91	0.82	0.61	80.4
DeepLabV3+ [124]	ResNet34	—	0.95	0.83	0.94	0.95	0.74	88.2
U-Net [124]	ResNet34	—	0.95	0.82	0.93	0.94	0.73	87.4
Our U-Net ResNet50	**ResNet50**	**32.6M**	**96.37**	**87.65**	**94.50**	**96.63**	**79.90**	**91.01**
Our Hybrid CNN–Transformer	**ResNet50+ViT**	**51.3M**	**96.41**	**87.02**	**94.60**	**96.61**	**78.27**	**90.58**

The Results in this table 5.18 are reported as Macro F1-Score (%). Values computed from per-class Precision/Recall in [124]. Our results on official validation split (1,602 images). Key observations for LandCover-AI:

- **No hybrid CNN–Transformer baseline exists** for LandCover-AI in the literature. To our knowledge, this work represents the first application of a TransUNet-style architecture to this dataset.
- **Our U-Net ResNet50 achieves state-of-the-art performance** with Macro F1 = **91.01%**, surpassing all previously reported CNN-based methods: DeepLabV3+ ResNet34 (88.2%, +2.81%), U-Net ResNet34 (87.4%, +3.61%), and vanilla U-Net (80.4%, +10.61%). This confirms the effectiveness of our composite Dice+Focal loss, multi-scale augmentation, and ResNet50 encoder on high-resolution aerial imagery.
- **The hybrid model achieves competitive results** (Macro F1 = 90.58%) despite being trained for only 24 epochs compared to 30 epochs for the CNN baseline. The marginal difference (−0.43% F1) is likely attributable to training time constraints rather than architectural limitations, as discussed in Section 5.6.2.7.
- **Class-wise analysis** (Table 5.17) reveals that both architectures excel on homogeneous, spectrally distinct classes (Water: >96% F1, Background: >96% F1). The CNN retains a slight advantage on narrow, edge-heavy classes (Road: +1.63% F1, Building: +0.63% F1), consistent with the theoretical strengths of local receptive fields for precise boundary detection.

Limitation note: The absence of published hybrid-model results on LandCover-AI prevents direct architecture-to-architecture comparison at equal training budget. Future work should establish standardized benchmarks for hybrid approaches on aerial land-cover datasets, reporting both F1-Score and mIoU for comprehensive evaluation.

5.6.2.7 Difficulties Encountered

The development of our semantic segmentation system for LandCover-AI encountered practical challenges primarily related to computational resources and training time constraints on cloud platforms.

1. **Hardware Limitations on Cloud Platforms:** Initial experiments on Google Colab were hindered by insufficient RAM (12–16 GB) and GPU memory fragmentation, causing frequent kernel restarts when processing 512×512 images

with batch size 8. We migrated to Kaggle, which provides 2× Tesla T4 GPUs (16 GB each) and 30 hours of weekly GPU access. Despite these improvements, GPU memory saturation during data loading required implementing a dynamic `tf.data` pipeline with on-the-fly augmentation and prefetching to avoid out-of-memory errors.

2. **Training Time Constraints and Architectural Trade-offs:** The hybrid CNN–Transformer architecture incurs significant computational overhead due to the self-attention mechanism ($\mathcal{O}(N^2)$ complexity with N tokens). On Kaggle's 12-hour session limit:

   - **U-Net ResNet50**: Completed 30 epochs in ≈6 hours 55 minutes , allowing full convergence and early-stopping recovery at epoch 28 (Macro F1 = 91.01%).
   - **Hybrid CNN-Transformer**: Completed only 24 epochs in ≈11h25m (≈28.5 min/epoch). Training was still improving at epoch 24 (validation F1 increased by +0.2% over the previous 3 epochs), but the session limit prevented further training.

   This disparity directly impacts the reported results: the hybrid model achieved 90.58% Macro F1 vs. 91.01% for the CNN, a difference of only 0.43 percentage points. Given that the hybrid model was still converging at epoch 24, we hypothesize that training for the full 30 epochs would likely close or reverse this gap. This highlights a critical trade-off: hybrid architectures offer theoretical advantages in global context modeling, but their practical deployment requires either extended training time or more powerful hardware.

5.7 Graphical User Interface

5.7.1 Overview

To make the results of this project accessible and interactive, we developed a web-based graphical user interface called **SATSEGMENT**. This application allows users to upload a satellite image and compare in real time the segmentation predictions produced by both models: the CNN-based U-Net ResNet50 and the hybrid CNN-Transformer TransUNet, across both datasets LoveDA and LandCover.

5.7.2 Technical Implementation

The application was implemented using the following technologies:

- **Backend:** Flask (Python), loading the saved `.keras` model files and performing preprocessing and inference on uploaded images.
- **Frontend:** HTML, CSS, and JavaScript providing a responsive dark-themed dashboard with real-time model switching and result rendering.

- **Inference Pipeline:** The uploaded image is resized to 512×512 pixels, normalized according to the dataset-specific preprocessing, and passed through the selected model. The output segmentation mask is colorized using the class palette and returned to the browser as an overlay image.

5.7.3 Interface Description

The SATSEGMENT dashboard is illustrated in Figure 5.32. The user can upload a satellite image in PNG, JPG, or TIF format, then select the dataset (LoveDA or LandCover) and click the **Compare Models** button to trigger both inferences simultaneously. The interface displays three panels side by side: the original uploaded image, the segmentation prediction of the CNN model (U-Net ResNet50), and the segmentation prediction of the hybrid model (TransUNet), allowing direct visual comparison between the two architectures.

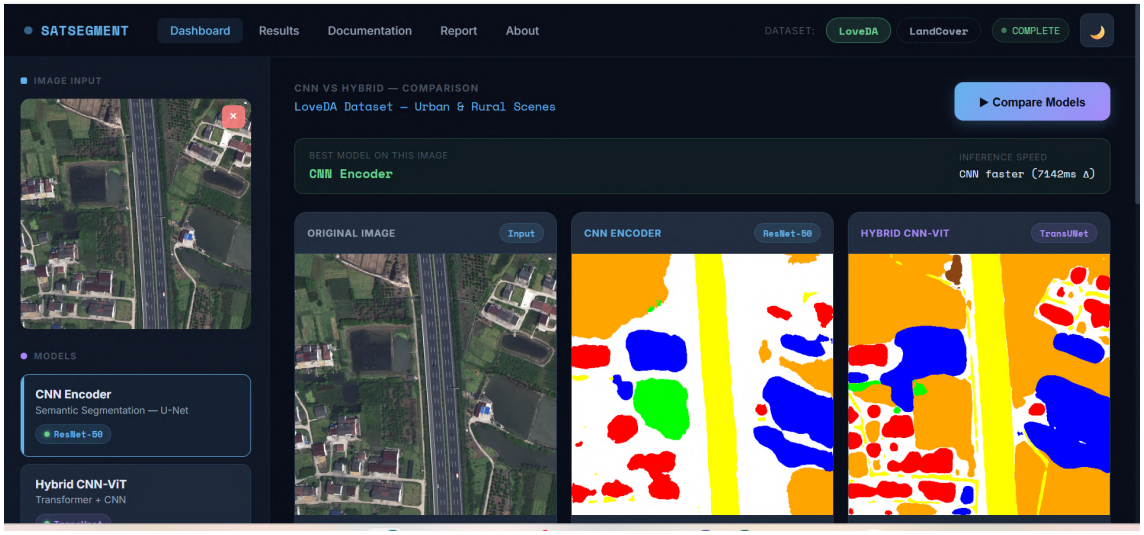


Figure 5.32: SATSEGMENT main dashboard showing the CNN vs Hybrid comparison on the LoveDA dataset.

The dashboard also provides a **Visual Analytics** section, illustrated in Figure 5.33, which displays for each model the class distribution of the predicted segmentation mask as a donut chart, along with the percentage of pixels assigned to each land-cover class. This allows the user to compare not only the visual quality of the predictions but also the quantitative class composition estimated by each architecture for the same input image.

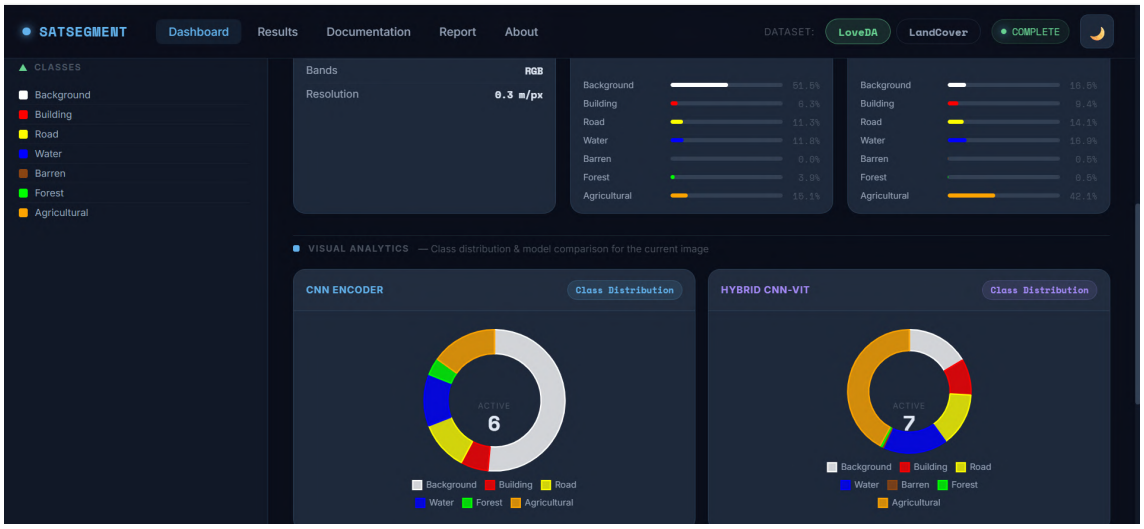


Figure 5.33: Visual Analytics section of the SATSEGMENT interface.

5.7.4 Documentation Interface

The SATSEGMENT application also includes a **Documentation** page, illustrated in Figure 5.34, which provides users with a detailed technical reference of the two model architectures. This section is organized as a structured sidebar navigation covering the following entries: Overview, Models (CNN Encoder and Hybrid CNN-ViT), Datasets, Metrics, and Usage Guide.

For each model, the documentation provides an interactive step-by-step visualization of the architecture pipeline.

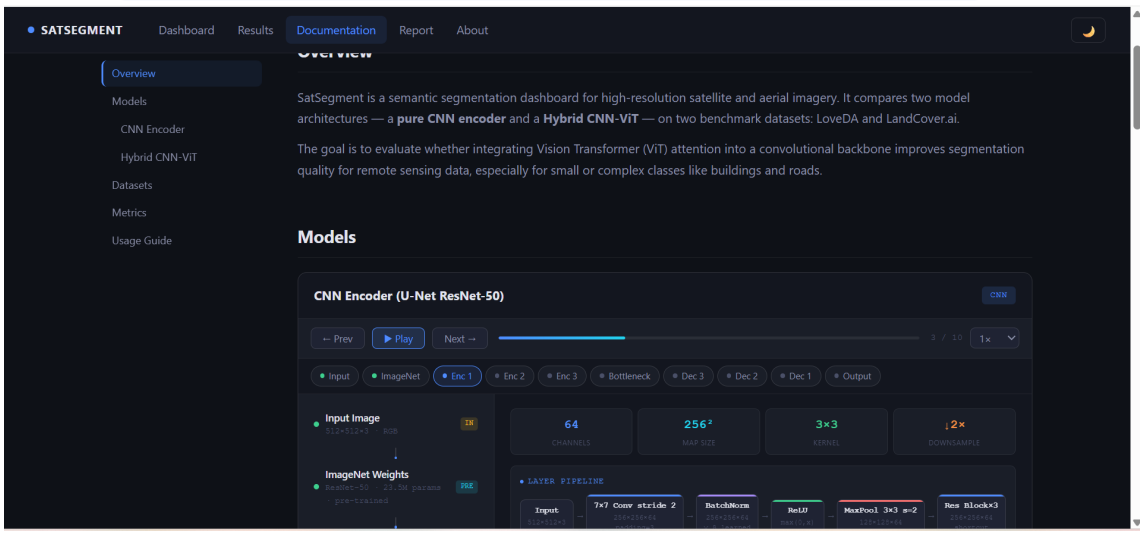


Figure 5.34: Documentation page of the SATSEGMENT interface.

5.8 Conclusion

Throughout the implementation of our semantic segmentation system for high-resolution satellite imagery, this chapter has highlighted several critical lessons learned across all stages of the development pipeline.

First, the **choice of dataset** proved to be a decisive factor. Working with two datasets of very different scales, the large-scale LoveDA dataset (≈9.58 GB, 5,987 images) and the more compact LandCover.ai dataset (≈2.1 GB, 10,674 tiles), allowed us to draw an important conclusion: dataset size and complexity directly condition the behaviour and the generalization capacity of Transformer-based architectures. On LoveDA, the TransUNet required a deliberate redistribution of the training pool (from the official 60/40 split to a proposed 80/20 split) to unlock its full potential, resulting in a significant improvement of +10.71% in mIoU (from 44.50% to 55.21%). This confirms that Transformers are fundamentally data-hungry and require a sufficiently large training set to learn meaningful global token relationships.

Second, the **architecture selection** revealed a clear trade-off between efficiency and global context modeling. The U-Net ResNet50, with 32.6M parameters, proved to be a strong and efficient baseline, achieving 50.69% mIoU on LoveDA and a state-of-the-art Macro F1 of 91.01% on LandCover.ai, with a training time of only ≈3–7 hours. The hybrid TransUNet ResNet50V2, with 51.3M parameters, demonstrated superior performance when given sufficient data, outperforming the CNN baseline on LoveDA (55.21% mIoU, +4.52%) while remaining competitive on LandCover.ai (90.58% Macro F1, −0.43%). This marginal gap on the smaller dataset is largely explained by training time constraints rather than architectural limitations.

Third, **preprocessing and data augmentation** played a central role in the quality of the learned representations. The combination of multi-scale resizing, random spatial cropping, geometric flipping, rotation, and color jitter provided the models with sufficiently diverse training examples to reduce overfitting and improve generalization across urban and rural scenes. Strict nearest-neighbor interpolation was enforced on all mask transformations to preserve label integrity at class boundaries.

Fourth, a careful **tuning of hyperparameters** was essential to stabilize training and maximize performance. The choice of optimizer (SGD with Nesterov momentum for the CNN; AdamW with gradient clipping for the Transformer), the composite loss functions (Dice + Focal for LoveDA; WCE + Dice for LandCover.ai), the learning rate schedules (Polynomial Decay and Warmup + Cosine Decay respectively), and the regularization strategies (Dropout, SpatialDropout2D, label smoothing, and partial backbone freezing) all contributed jointly to achieving reliable and reproducible results across both datasets and both architectures.

Finally, to make these results accessible and interactive, we developed the **SAT-SEGMENT** web application, which allows users to upload a satellite image, select a dataset, and visually compare in real time the segmentation predictions produced by both models side by side, together with a quantitative per-class pixel distribution analysis and a full documentation of the architectures.

Overall, the results obtained confirm that the hybrid CNN-Transformer paradigm represents the most effective strategy for semantic segmentation of high-resolution

satellite imagery when sufficient training data is available, while the pure CNN baseline remains a competitive and resource-efficient alternative for constrained environments.

General Conclusion

In recent years, the rapid evolution of Artificial Intelligence has completely transformed the field of remote sensing, turning massive streams of raw satellite imagery into a powerful source for understanding our changing planet. Traditional geographic mapping methods, which once relied heavily on manual interpretation or rigid pixel-matching algorithms, are no longer sufficient to handle the vast amount of high-resolution spatial data available today. Modern deep learning architectures have stepped in to bridge this gap, offering the unique ability to automatically detect, classify, and understand complex surface features at a scale never seen before. It is within this exciting intersection of AI and Earth observation that this thesis was established.

The main goal of this project was to tackle one of the most vital challenges in satellite remote sensing: accurately mapping complex terrains from space using semantic segmentation. Because high-resolution satellite imagery contains highly detailed surface features, shifting lighting conditions, and diverse geographic landscapes, traditional deep learning models often struggle to get class boundaries perfectly right. To solve this, we implemented and evaluated a hybrid CNN–Transformer architecture, which successfully combines the precise, localized detailing of Convolutional Neural Networks (CNNs) with the big-picture context awareness of Vision Transformers.

Our work followed a step-by-step engineering approach. We first built a classic baseline utilizing a U-Net architecture with a ResNet50 encoder. While reliable, we noticed its limitations in understanding long-range spatial context. This directly led us to implement a more advanced hybrid model, TransUNet (equipped with a ResNet50V2 encoder), which effectively bridges the gap between local boundary details and a global understanding of the overall scene.

To thoroughly test the robustness of our models, extensive experiments were carried out using the Kaggle platform across two benchmark datasets: LoveDA (focusing on domain shifts between urban and rural environments) and LandCover-AI (demanding high precision for infrastructure classes). Our experimental findings yielded critical insights. While the standard U-Net baseline performed well on massive, distinct regions like forests and water bodies, it frequently caused pixel fragmentation along tricky borders. In contrast, the hybrid model demonstrated superior spatial consistency, keeping boundaries clean and significantly reducing thematic confusion. Furthermore, our tests proved that moving away from heavily skewed data splits to our proposed balanced 80/20 data partition strategy brought an im-

mediate, noticeable boost in per-class accuracy across the diagonal of the confusion matrices.

Finally, to bridge the gap between academic research and a functional application, we successfully built a professional graphical user interface, the **SATSEGMENT** Dashboard. Featuring both light and dark mode viewports along with interactive documentation, this interface seamlessly handles the deployment of our models for real-world visualization.

While the results of this thesis prove that hybrid models work incredibly well for high-resolution satellite images, there is one major path we want to explore next: **Multi-Modal Data Fusion**. Right now, our model relies on standard optical (RGB) pictures, which means it cannot see through clouds, smoke, or bad weather. In the future, we can fix this by combining these regular images with Radar (SAR) or multi-spectral data. This mix would give the system "all-weather vision," allowing it to monitor the Earth continuously, day or night, rain or shine.

In conclusion, this project successfully demonstrates that merging local convolutions with global attention mechanisms provides an exceptional framework for automated geographic mapping. By combining advanced deep learning engineering with a practical dashboard interface, this work provides a solid foundation for the next generation of smart Earth observation tools.

Bibliography

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, p. 52, 2023.

[3] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat, "Cnn variants for computer vision: History, architecture, application, challenges and future scope," *Electronics*, vol. 10, no. 20, p. 2470, 2021.

[4] M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad, and M. Khan, "Comparative analysis of recent architecture of convolutional neural network," *Mathematical Problems in Engineering*, vol. 2022, no. 1, p. 7313612, 2022.

[5] N. Patwardhan, S. Marrone, and C. Sansone, "Transformers in the real world: A survey on nlp applications," *Information*, vol. 14, no. 4, p. 242, 2023.

[6] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[7] A. A. Aleissaee, A. Kumar, R. M. Anwer, S. Khan, H. Cholakkal, G.-S. Xia, and F. S. Khan, "Transformers in remote sensing: A survey," *Remote Sensing*, vol. 15, no. 7, p. 1860, 2023.

[8] J. Jakubik, S. Roy, C. Phillips, P. Fraccaro, D. Godwin, B. Zadrozny, D. Szwarcman, C. Gomes, G. Nyirjesy, B. Edwards *et al.*, "Foundation models for generalist geospatial artificial intelligence," *arXiv preprint arXiv:2310.18660*, 2023.

[9] G. Wang, H. Chen, L. Chen, Y. Zhuang, S. Zhang, T. Zhang, H. Dong, and P. Gao, "P 2fevit: plug-and-play cnn feature embedded hybrid vision transformer for remote sensing image classification," *Remote Sensing*, vol. 15, no. 7, p. 1773, 2023.

[10] Q. Zhang, X. Hu, and Y. Xiao, "A novel hybrid model based on cnn and multi-scale transformer for extracting water bodies from high resolution remote sensing images," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 10, pp. 889–894, 2023.

[11] X. Chen, D. Li, M. Liu, and J. Jia, "Cnn and transformer fusion for remote sensing image semantic segmentation," *Remote Sensing*, vol. 15, no. 18, p. 4455, 2023.

[12] R. Wang, L. Ma, G. He, B. A. Johnson, Z. Yan, M. Chang, and Y. Liang, "Transformers for remote sensing: A systematic review and analysis," *Sensors*, vol. 24, no. 11, p. 3495, 2024.

[13] L. Wu, R. Liu, N. Ju, A. Zhang, J. Gou, G. He, and Y. Lei, "Landslide mapping based on a hybrid cnn-transformer network and deep transfer learning using remote sensing images with topographic and spectral features," *International Journal of Applied Earth Observation and Geoinformation*, vol. 126, p. 103612, 2024.

[14] Y. Huang, D. Jiao, X. Huang, T. Tang, and G. Gui, "A hybrid cnn-transformer network for object detection in optical remote sensing images: integrating local and global feature fusion," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2024.

[15] M. Yao, Y. Zhang, G. Liu, and D. Pang, "Ssnet: A novel transformer and cnn hybrid network for remote sensing semantic segmentation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 3023–3037, 2024.

[16] J. Li, J. Zhang, and Y. Fu, "Cthnet: A cnn–transformer hybrid network for landslide identification in loess plateau regions using high-resolution remote sensing images," *Sensors*, vol. 25, no. 1, p. 273, 2025.

[17] T. He, J. Chen, and D. Pan, "Gofenet: A hybrid transformer–cnn network integrating geobia-based object priors for semantic segmentation of remote sensing images," *Remote Sensing*, vol. 17, no. 15, p. 2652, 2025.

[18] C. Elachi and J. J. Van Zyl, *Introduction to the physics and techniques of remote sensing*. John Wiley & Sons, 2021.

[19] T. M. Lillesand, R. W. Kiefer, and J. W. Chipman, *Remote Sensing and Image Interpretation*, 7th ed. Hoboken, NJ: Wiley, 2015.

[20] Faculté des Sciences et de la Technologie, Université Mohamed El Bachir El Ibrahimi, *Cours de Télédétection*, 2023, available online: https://fac.umc.edu.dz/fst/fichiers/TélédétectionCours.pdf. [Online]. Available: https://fac.umc.edu.dz/fst/fichiers/TélédétectionCours.pdf

[21] L. Part, "Photogrammetry and remote sensing," 2023.

[22] P. K. Meena, "1pooja kumari meena and 2anchal karol," *Sustainable Agriculture in the 21st Century*, p. 71, 2024.

[23] R. Mukherjee, "Basics of remote sensing."

[24] F. Dulam, P. Solanki, M. Yadav, K. V. Pandey, R. Rajput, M. Kumar, A. P. Singh, and S. Srishtty, "Applications of remote sensing in horticulture: a review," *Plant Archives*, vol. 25, no. 1, pp. 2552–2561, 2025.

[25] P. Garg, *Remote Sensing: Theory and Applications*. Walter de Gruyter GmbH & Co KG, 2024.

[26] S. Mandloi and J. Bangre, "Modern applications of remote sensing in agriculture," *AGRONOMY*, p. 1, 2024.

[27] F. E. Fassnacht, J. C. White, M. A. Wulder, and E. Næsset, "Remote sensing in forestry: current challenges, considerations and directions," *Forestry: An International Journal of Forest Research*, vol. 97, no. 1, pp. 11–37, 2024.

[28] G. E. Adjovu, H. Stephen, D. James, and S. Ahmad, "Overview of the application of remote sensing in effective monitoring of water quality parameters," *Remote Sensing*, vol. 15, no. 7, p. 1938, 2023.

[29] J. Jung, M. Maeda, A. Chang, M. Bhandari, A. Ashapure, and J. Landivar-Bowles, "The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems," *Current Opinion in Biotechnology*, vol. 70, pp. 15–22, 2021.

[30] S. Jadhav, M. Durairaj, R. Reenadevi, R. Subbulakshmi, V. Gupta, and J. V. N. Ramesh, "Spatiotemporal data fusion and deep learning for remote sensing-based sustainable urban planning," *International Journal of System Assurance Engineering and Management*, pp. 1–9, 2024.

[31] Y. Mahajan, A. P. Singh, I. Singh, S. Yadav, A. Aneja, and M. Vaqui, "Fundamentals of active and passive microwave remote sensing: principles," *RADAR: Remote Sensing Data Analysis with Artificial Intelligence*, p. 31, 2025.

[32] D. D. Champaneri, K. D. Desai, T. R. Ahlawat, and N. K. Pampaniya, "Transforming fruit farming in a hi-tech way through remote sensing: A review," in *Biological Forum–An International Journal*, vol. 15, no. 2, 2023, pp. 333–341.

[33] UVED, "Envcal – environmental monitoring by remote sensing, online course," Université Paris 1 Panthéon-Sorbonne, 2008, available as part of the Environmental Monitoring curriculum. [Online]. Available: https://www.uved.fr/

[34] S. Aggarwal, "Principles of remote sensing," *Satellite remote sensing and GIS applications in agricultural meteorology*, vol. 23, no. 2, pp. 23–28, 2004.

[35] M. A. Shareef, "Introduction to remote sensing."

[36] GIS Geography, "Energy interaction in remote sensing: Reflection, absorption, and transmission," 2024, accessed: April 20, 2026. [Online]. Available: https://gisgeography.com/energy-interaction-remote-sensing-light-reflection-absorption-transmission/

[37] J. M. Rajab, "Conservation of energy and radiation interactions," https://uomustansiriyah.edu.iq/media/lectures/6/6_2022_10_20!07_53_58_PM.pdf, n.d.

[38] Natural Resources Canada, "Fundamentals of remote sensing," 2024, consulted on April 20, 2026. [Online]. Available: https://natural-resources.canada.ca/sites/nrcan/files/earthsciences/pdf/resource/tutor/fundam/pdf/fundamentals_e.pdf

[39] S. Tripathi, B. K. Gupta, S. Awasthi, A. Pandey, and B. P. Mishra, "Remote sensing: Principles, techniques, and applications," in *Modern Technologies in Sustainable Farming*. Integrated Publication, 2023, ch. 5, pp. 65–80.

[40] M. V. K. Sivakumar, P. S. Roy, K. Harmsen, and S. K. Saha, "Satellite remote sensing and gis applications in agricultural meteorology," in *Proceedings of the Training Workshop*. Dehra Dun, India: World Meteorological Organization (WMO), 2003. [Online]. Available: https://www.preventionweb.net/files/1682_9970.pdf?startDownload=true

[41] E. D. Conway, *An introduction to satellite image interpretation*. JHU Press, 1997.

[42] A. Shrivastava, "Panchromatic imagery: High-resolution mapping explained," https://www.geowgs84.com/post/panchromatic-imagery-high-resolution-mapping-explained, GeoWGS84, September 2025, accessed: 2026-04-30.

[43] Esri, "Multispectral image definition | gis dictionary," https://support.esri.com/en-us/gis-dictionary/multispectral-image, Esri, n.d., accessed: 2026-04-30.

[44] J. M. Amigo, "Chapter 1.1 - hyperspectral and multispectral imaging: setting the scene," in *Hyperspectral Imaging*, ser. Data Handling in Science and Technology, J. M. Amigo, Ed. Elsevier, 2019, vol. 32, pp. 3–16. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444639776000018

[45] Y. Zhang, D. Sidibé, O. Morel, and F. Mériaudeau, "Deep multimodal fusion for semantic image segmentation: A survey," *Image and Vision Computing*, vol. 105, p. 104042, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885620301748

[46] GeoWGS84, "What is sar imagery: A complete guide to synthetic aperture radar," https://www.geowgs84.com/post/what-is-sar-imagery-a-complete-guide-to-synthetic-aperture-radar, n.d.

[47] GeoServe. (2026) Optical imagery - fundamentals and sensor types. Accessed: April 10, 2026. [Online]. Available: https://geoserve.nl/optical-imagery/

[48] Spatialnode, "Satellite imagery: An overview," 2023, accessed: 2026-04-10. [Online]. Available: https://www.spatialnode.net/articles/understanding-satellite-imagery50384b

[49] NASA Earthdata, "Remote sensing resolution," 2024, accessed: 2026-04-10. [Online]. Available: https://www.earthdata.nasa.gov/learn/earth-observation-data-basics/remote-sensing-resolution

[50] Belgian Platform on Earth Observation (BELSPO), "Temporal resolution," 2023, accessed: 2026-04-10. [Online]. Available: https://eo.belspo.be/en/iii23-temporal-resolution

[51] W. B. Green, D. Jensen, and A. Culver, "Digital processing of remotely sensed imagery," in *Review of Progress in Quantitative Nondestructive Evaluation: Volume 17A*. Springer, 1998, pp. 23–31.

[52] Data Science UA, "Image processing techniques, image types and applications," Data Science UA Blog, 2024, accessed: April 19, 2026. [Online]. Available: https://data-science-ua.com/blog/image-processing-techniques-image-types-and-applications/

[53] Lenovo, "What is grayscale?" Lenovo Glossary, 2024, accessed: April 19, 2026. [Online]. Available: https://www.lenovo.com/us/en/glossary/grayscale/

[54] A. Pinhero, M. Anupama, P. Vinod, C. A. Visaggio, N. Aneesh, S. Abhijith, and S. AnanthaKrishnan, "Malware detection employed by visualization and deep neural network," *Computers & Security*, vol. 105, p. 102247, 2021.

[55] S. Suyarso, M. D. Setiawati, I. H. Supriyadi, and B. Prayudha, "Climate change indicator, impact, adaptation, and innovation at the local level: learn from the peoples' experience of the coastal plain of probolinggo, east java, indonesia," in *Climate change, community response and resilience*. Elsevier, 2023, pp. 93–118.

[56] S. S. Baboo and M. R. Devi, "Geometric correction in recent high resolution satellite imagery: a case study in coimbatore, tamil nadu," *International Journal of Computer Applications*, vol. 14, no. 1, pp. 32–37, 2011.

[57] G. Geography, "What is atmospheric correction in remote sensing," 2021.

[58] P. Mini and P. A. Bhomini, "A study in enhancement of satellite images," *Engineering Applications of Artificial Intelligence*, vol. 138, p. 109259, 2024.

[59] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3523–3542, 2022.

[60] M. Thoma, "A survey of semantic segmentation," *arXiv preprint arXiv:1602.06541*, 2016. [Online]. Available: https://arxiv.org/abs/1602.06541

[61] F. Sultana, A. Sufian, and P. Dutta, "Evolution of image segmentation using deep convolutional neural network: A survey," *arXiv preprint arXiv:2001.04074*, 2020.

[62] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, "Panoptic segmentation," in *CVPR*, 2019.

[63] J. Wang, Z. Zheng, A. Ma, X. Lu, and Y. Zhong, "Loveda: A remote sensing land-cover dataset for domain adaptive semantic segmentation," *arXiv preprint arXiv:2110.08733*, 2021.

[64] B. Neupane, T. Horanont, and J. Aryal, "Deep learning-based semantic segmentation of urban features in satellite images: A review and meta-analysis," *Remote Sensing*, vol. 13, no. 4, p. 808, 2021.

[65] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[66] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[67] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[68] S. Horowitz and T. Pavlidis, "Picture segmentation by a directed split and merge procedure," *Proceedings of the 2nd IAPR*, pp. 424–433, 1976.

[69] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[70] D. Bradley and G. Roth, "Adaptive thresholding using the integral image," *Journal of Graphics Tools*, vol. 12, no. 2, pp. 13–21, 2007.

[71] Stanford Institute for Human-Centered AI. (2020) Artificial intelligence: Definitions and perspectives. Accessed February 20, 2025. [Online]. Available: https://hai-production.s3.amazonaws.com/files/2020-09/AI-Definitions-HAI.pdf

[72] M. Pichler and F. Hartig, "Machine learning and deep learning—a review for ecologists," *Methods in Ecology and Evolution*, vol. 14, no. 4, pp. 994–1016, 2023, figure 2: Relationship between artificial intelligence (AI), machine learning (ML), and deep learning (DL).

[73] Wikipedia Contributors. (2025) Expert system. Accessed February 20, 2025. [Online]. Available: https://en.wikipedia.org/wiki/Expert_system

[74] IBM. (2024) The history of artificial intelligence. Accessed February 20, 2025. [Online]. Available: http://ibm.com/think/topics/history-of-artificial-intelligence

[75] GeeksforGeeks. (2024) Difference between human expert and expert system. Accessed February 20, 2025. [Online]. Available: https://www.geeksforgeeks.org/dbms/difference-between-human-expert-and-expert-system/

[76] Scribd. (2019) Mycin: Computer-based medical consultations. Accessed February 20, 2025. [Online]. Available: https://fr.scribd.com/document/410324333/Mycin

[77] Python in Plain English. (n.d.) Various types of machine learning analysis. Accessed February 20, 2025. [Online]. Available: https://python.plainenglish.io/various-types-of-machine-learning-analysis-a6e027a62db4

[78] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2009, accessed February 20, 2025. [Online]. Available: https://www.molgen.mpg.de/3659531/MITPress--SemiSupervised-Learning.pdf

[79] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018, accessed February 20, 2025. [Online]. Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[80] M. L. Benomar, "Deep learning (dl) [course lecture notes]," Unpublished course material, Ain Temouchent, Algeria, 2024, academic year 2024–2025.

[81] Levity. (2023, March) Deep learning vs machine learning: What's the difference? Accessed February 20, 2025. [Online]. Available: https://medium.com/levity/deep-learning-vs-machine-learning-whats-the-difference-e367803bb96d

[82] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[83] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[84] M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, p. 151, 2023.

[85] X. Liu, F. Han, K. H. Ghazali, I. I. Mohamed, and Y. Zhao, "A review of convolutional neural networks in remote sensing image," in *Proceedings of the 2019 8th international Conference on software and computer applications*, 2019, pp. 263–267.

[86] T. Bezdan, "Convolutional neural network layers and architectures," in *Proceedings of the International Scientific Conference-Sinteza 2019*, 2019.

[87] S. Lambert. (2024) Cs230: Convolutional neural networks cheatsheet. Accessed February 20, 2025. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks/#convolution

[88] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," *arXiv preprint arXiv:2009.07485*, 2020, accessed February 20, 2025. [Online]. Available: https://arxiv.org/pdf/2009.07485

[89] GeeksforGeeks. (2024) Activation functions in neural networks. Accessed February 20, 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/activation-functions-neural-networks/

[90] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[91] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[92] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

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

[94] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241.

[95] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[96] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[97] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[98] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," vol. 40, no. 4. IEEE, 2017, pp. 834–848.

[99] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[100] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2117–2125.

[101] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[102] GeeksforGeeks. (2024) Vision transformer (vit) architecture. Accessed February 20, 2025. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/vision-transformer-vit-architecture/

[103] X. Li. (2021) Paper reading summary #1: Vision transformer. Accessed February 20, 2025. [Online]. Available: https://medium.com/@lixue421/paper-reading-summary-1-vision-transformer-1489da75ea0b

[104] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[105] X. Guan, H. Yang, H. Sun, P. Zhu, H. Yu, and K. Xia, "Medfreq-net: Medical frequency network with hybrid cnn-transformer and enhanced features for 3d medical image segmentation," *Alexandria Engineering Journal*, 2024.

[106] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *arXiv preprint arXiv:1911.02685*, 2019, accessed February 20, 2025. [Online]. Available: https://arxiv.org/pdf/1911.02685.pdf

[107] Stanford University. (2024) Cs231n: Convolutional neural networks for visual recognition – transfer learning. Accessed February 20, 2025. [Online]. Available: https://cs231n.github.io/transfer-learning/

[108] N. Mungoli, "Exploring the limits of deep learning: A study on overfitting and regularization techniques," *International Journal of Advanced Engineering Technologies and Innovations*, vol. 1, no. 4, pp. 1–18, 2023. [Online]. Available: https://media.neliti.com/media/publications/603604-exploring-the-limits-of-deep-learning-a-1bf40ade.pdf

[109] T. Hayashi, T. Shimizu, and Y. Fukami, "Collaborative problem solving on a data platform kaggle," *arXiv preprint arXiv:2107.11929*, 2021.

[110] Y. Dong, "Beating kaggle the easy way," 2015.

[111] K. Banachewicz and L. Massaron, *The Kaggle Book: Data analysis and machine learning for competitive data science*. Packt Publishing Ltd, 2022.

[112] K. Bönisch and L. Losaria, "Kaggle chronicles: 15 years of competitions, community and data science innovation," *arXiv preprint arXiv:2511.06304*, 2025.

[113] C. Tauchert, P. Buxmann, and J. Lambinus, "Crowdsourcing data science: A qualitative analysis of organizations' usage of kaggle competitions," 2020.

[114] K. Srinath, "Python–the fastest growing programming language," *International Research Journal of Engineering and Technology*, vol. 4, no. 12, pp. 354–357, 2017.

[115] G. Van Rossum and F. L. Drake Jr, *El tutorial de Python*. Python Software Foundation, 2017.

[116] P. Singh and A. Manure, *Learn TensorFlow 2.0: Implement machine learning and deep learning models with Python*. Apress, 2019.

[117] A. S. Saabith, T. Vinothraj, and M. Fareez, "Popular python libraries and their application domains," *International Journal of Advance Engineering and Research Development*, vol. 7, no. 11, 2020.

[118] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[119] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[120] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[121] A. Boguszewski, D. Batorski, N. Ziemba-Jankowska, T. Dziedzic, and A. Zambrzycka, "Landcover.ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 1102–1110.

[122] ——, "Landcover. ai: Dataset for automatic mapping of buildings, woodlands, water and roads from aerial imagery," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1102–1110.

[123] L. Wang, R. Li, C. Zhang, S. Fang, C. Duan, X. Meng, and P. M. Atkinson, "Unetformer: A unet-like transformer for efficient semantic segmentation of remote sensing urban scene imagery," *arXiv preprint arXiv:2109.08937*, 2021. [Online]. Available: https://arxiv.org/abs/2109.08937

[124] E. S. Zamanoglu, S. Kosunalp, S. Erbay, V. Tumen, E. Cengil, and K. Demir, "Land cover segmentation using DeepLabV3 and ResNet50," in *2023 4th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)*. Plovdiv, Bulgaria: IEEE, November 2023, pp. 1–5.