HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Capstone Project

Road Segmentation for Aerial Images

Student: *Phùng Tiến Thành - 20225531*
Nguyễn Phan Thắng - 20225529
Đinh Bảo Hưng - 20225446

Class: *157213*

Lecturer: *Ph. D. Trần Nguyên Ngọc*

Hanoi, June 2025

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to Dr. Tran Nguyen Ngoc, our esteemed Computer Vision professor, whose guidance and expertise were instrumental in the successful completion of this aerial road segmentation project. As our dedicated instructor and mentor, Dr. Ngoc not only provided us with a profound understanding of image segmentation techniques but also inspired us to explore the challenges of road extraction through his insightful lectures and hands-on advice. His thoughtful suggestions—from dataset selection to model architecture—helped us navigate technical hurdles and refine our approach at every stage. We are deeply inspired by his passion for research, commitment to excellence, and patience in mentoring, which embody the qualities of an exceptional educator.

Our sincere thanks also go to the creators of the DeepGlobe and Massachusetts Roads Dataset, and the open-source community for sharing valuable resources, benchmarks, and code implementations. Their contributions enabled us to build upon existing work and focus on innovation.

Special appreciation goes to our project team members for their collaborative spirit, late-night debugging sessions, and relentless efforts to tackle challenges together. This project has not only honed our technical skills in deep learning and computer vision but also strengthened our teamwork and problem-solving abilities.

Lastly, we hope to have future opportunities to learn from Dr. Ngoc and collaborate with all contributors again. The knowledge gained from this project will serve as a foundation for our continued exploration of AI applications in remote sensing and urban planning. We are truly grateful for this rewarding academic experience.

TABLE OF CONTENTS

CHAPTER I. INTRODUCTION

1.1 Background and Motivation

Semantic segmentation has become a foundational task in the field of computer vision, with applications ranging from medical image analysis to autonomous driving and satellite imagery interpretation. Among these, road segmentation from aerial and satellite imagery plays a critical role in geographic information systems (GIS), urban planning, and navigation technologies.

Manual annotation of roads is not only time-consuming and labor-intensive but also prone to human error. Hence, automating this process using deep learning methods has become increasingly appealing. However, road segmentation poses several challenges: roads are often thin, winding, and can be partially occluded by shadows, trees, or buildings. This requires models capable of precise localization and high spatial resolution to effectively distinguish roads from similar-looking background elements.

1.2 Problem Statement

The task addressed in this study is binary road segmentation in high-resolution satellite images. Specifically, the objective is to develop a model that can take a satellite image as input and output a binary mask, where pixels belonging to roads are labeled as 1 and all other pixels as 0. The goal is to accurately capture road structures, including small rural paths and complex urban networks.

1.3 Existing Approaches

A wide range of convolutional neural network (CNN)-based models have been proposed for semantic segmentation, including Fully Convolutional Networks (FCNs), U-Net, DeepLabV3+, and more recently, Transformer-based architectures like SegFormer. These models have shown promising results in general-purpose segmentation tasks.

However, they often struggle in road segmentation due to the linear and small-scale nature of roads, which can be overlooked in deeper layers of CNNs. Models lacking skip connections or multi-scale context aggregation may fail to retain spatial precision, resulting in fragmented or blurry predictions.

1.4 Proposed Approaches

To address the challenges mentioned above, we adopt **D-LinkNet**, a segmentation model specifically designed for road extraction. D-LinkNet combines a strong encoder (ResNet34) with a carefully designed decoder and dilated convolution blocks (DBlocks) to capture both global context and fine spatial details. One key strength of D-LinkNet is the use of **skip connections**, which enable the model to retain high-resolution features from early layers. This proves especially beneficial in reconstructing narrow road structures.

The training pipeline includes extensive preprocessing and augmentation strategies, as well as robust training techniques such as **BCE + Dice loss**, **AdamW optimizer**, **ReduceLROnPlateau scheduler**, and **early stopping**. These components collectively help the model generalize better and avoid overfitting.

In addition to D-LinkNet, we also **fine-tune a SegFormer-B1 model**, a modern transformer-based architecture known for its strong semantic segmentation performance. SegFormer replaces convolutional encoders with a **MiT (Mix Vision Transformer)** backbone, enabling the model to capture long-range dependencies and global context more effectively. Unlike traditional CNN-based models, SegFormer does not rely on heavy decoders. Instead, it uses lightweight MLP decoders with multi-scale features, making it both **computationally efficient** and **context-aware**.

To adapt SegFormer for our binary road segmentation task, we fine-tune a pre-trained `nvidia/segformer-b1-finetuned-ade-512-512` model on our dataset. The original decoder head is adjusted to output a single channel, and the output logits are upsampled via bilinear interpolation to match the input mask resolution. The same training strategy (BCE + Dice loss, AdamW, scheduler, early stopping) is applied to ensure a fair comparison with D-LinkNet.

By incorporating both convolution-based and transformer-based architectures, our approach explores **complementary modeling capabilities**. While D-LinkNet excels at preserving fine spatial details through skip connections and high-resolution decoding, SegFormer offers stronger **global reasoning** and **generalization** across varied road topologies.

1.5 Contributions

The main contributions of this project are as follows:

- Integration and harmonization of two diverse datasets: Massachusetts Roads and DeepGlobe Road Extraction.
- Development of a modular and efficient preprocessing pipeline that avoids redundant data folders and supports dynamic resizing.
- Building D-LinkNet (using pretrained weights for the encoder, and the decoder weights were initialized from scratch) with carefully designed training strategies, including normalization, data augmentation, and custom loss functions.
- Comparative integration of SegFormer-B1 as a transformer-based alternative to D-LinkNet, highlighting architectural diversity and segmentation performance.
- In-depth evaluation using both quantitative metrics (Dice, IoU) and qualitative visualizations.
- Future extensibility for incorporating other models (e.g., DeepLabV3+) and building a user-facing demo application.

CHAPTER II. DATA PREPROCESSING

Data preprocessing is a fundamental step in any machine learning pipeline, especially in computer vision tasks such as semantic segmentation. Its goal is to ensure that all inputs to the model are consistent, clean, and structured in a way that maximizes the learning capacity of the network. In this project, preprocessing involved several key stages: preparing the dataset, processing images and masks, applying data augmentation techniques, and building a reliable data loading pipeline.

2.1 Dataset Overview

The dataset for our project comes from two primary sources: the **Massachusetts Roads Dataset** [1] and the **DeepGlobe Road Extraction Dataset** [2]. Both datasets consist of high-resolution satellite imagery paired with annotated binary masks indicating road locations. Fortunately, both datasets adopted the same labeling annotation: white pixels representing roads and black pixels representing background. This consistency helped us avoid an additional label standardization step, simplifying the integration process and ensuring direct compatibility between the two sources.

The **Massachusetts Roads Dataset** contains images at various resolutions and urban contexts, offering clear delineation of road networks primarily in the northeastern United States. The dataset was provided in `.tiff` and `.tif` format with separate folders for training, validation, and test images, alongside their corresponding ground-truth masks.

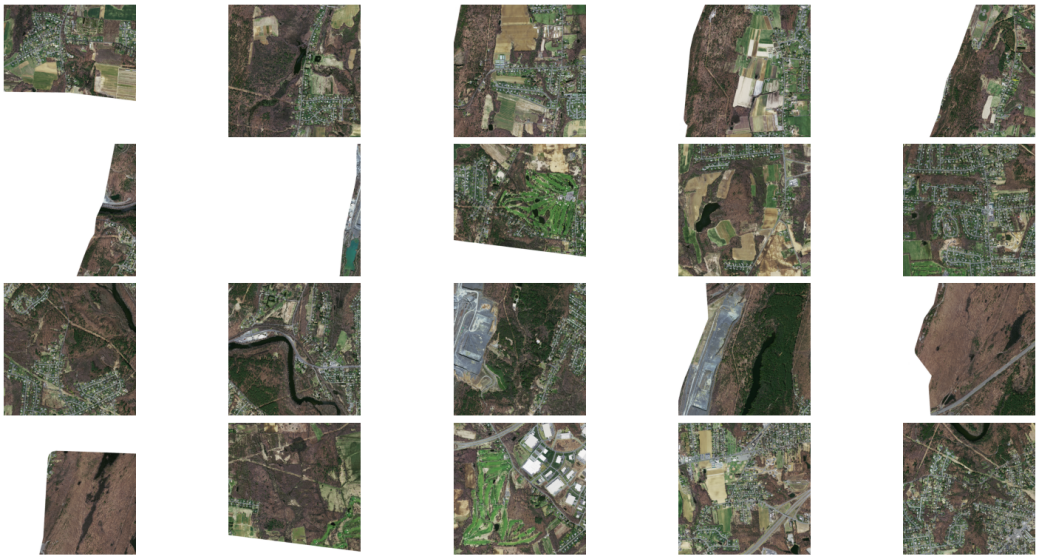

Figure 1 Some images in the Massachusetts dataset

The **DeepGlobe Road Extraction Dataset**, on the other hand, features a global collection of satellite images with diverse geographical coverage. Its images are uniformly sized at 1024×1024 pixels and are paired with masks labeled in `.jpg` and `.png` format.

Figure 2 Some images in the DeepGlobe dataset

Initially, a third dataset—the **Inria Aerial Image Labeling Dataset**—was considered. However, upon visual inspection, we determined that this dataset predominantly focused on building segmentation, with mask annotations more suitable for detecting structures rather than roads, some of the masks are shown in the figure below. Moreover, its visual style and annotation patterns differed significantly from Massachusetts and DeepGlobe, posing challenges for model consistency and generalization in road detection tasks. Therefore, the Inria dataset was excluded from the training pipeline.

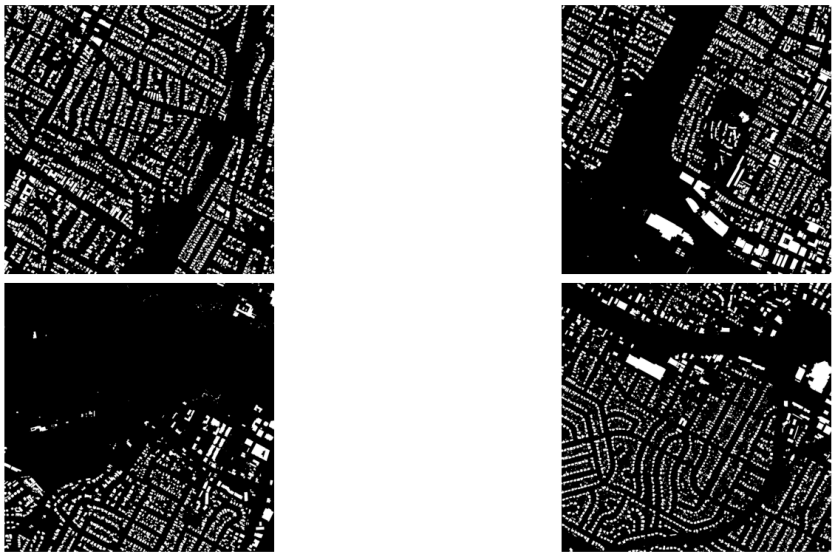

Figure 3 The masks in the Inria dataset

To create a unified dataset, the Massachusetts and DeepGlobe images and masks were consolidated into a single dataset directory. All files were converted to `.tiff` format to standardize storage and facilitate streamlined loading across different libraries.

The consolidated dataset was split into a ratio of 70% - 15% - 15%, respectively, for training, validation, and testing, ensuring no overlap between splits. The splitting scheme was explicitly documented in a `split.csv` file, mapping each image filename to its designated partition for reproducibility.

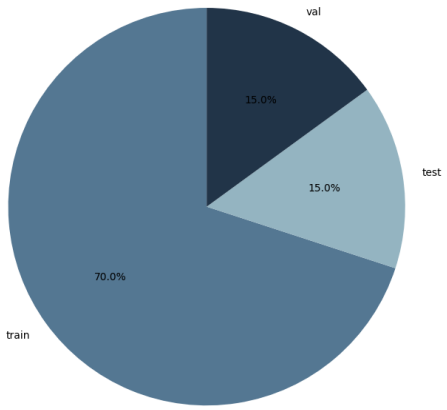

Figure 4 Dataset Split Distribution

2.2 Image and Mask Processing

Once the datasets were merged into a unified collection, we performed several preprocessing steps on both the input images and their corresponding segmentation masks to ensure compatibility with the model and improve learning stability.

2.2.1 Image Preprocessing

The original images varied in resolution across the datasets. In order to standardize the input size for the neural network, all images were **resized to 1024×1024 pixels** (for DLinkNet), and **512×512 pixels** (for SegFormer-B1), using bilinear interpolation. These sizes were chosen as a balance between preserving sufficient spatial detail of road networks while remaining computationally tractable for training on modern GPUs with limited memory.

Following resizing, the images were normalized to enhance convergence during training. Specifically, pixel values, originally ranging from [0, 255], were scaled to [0, 1] by dividing by 255.0. Subsequently, a custom normalization was applied.

For DLinkNet:

$$I_{\text{norm}} = I \times 3.2 - 1.6$$

where I represents the scaled image tensor.

Figure 5 An image before and after applying the normalization formula above

This normalization formula was adopted from a previously published implementation available at [3], where it was empirically shown to enhance training stability for road extraction tasks. By expanding the dynamic range and centering the data

around zero, this approach better aligns the input distribution with the pretraining statistics of the ResNet34 backbone used in our model.

For SegFormer-B1:

$$I_{\text{norm}} = I \times 2 - 1$$

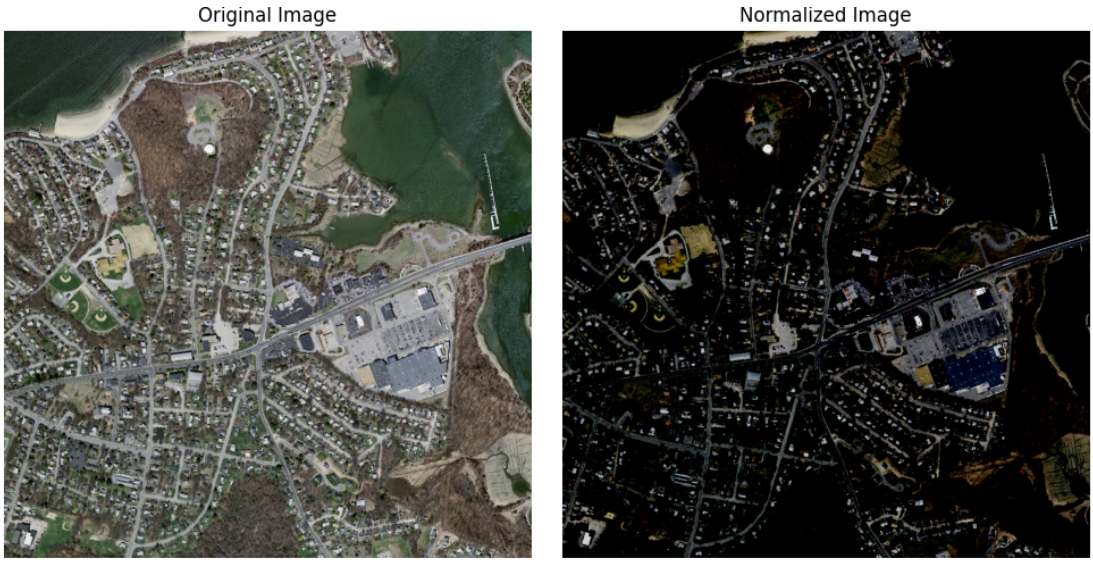

Figure 6 An image before and after applying the normalization formula above

The normalization formula used for SegFormer-B1 was chosen to align the input data with the expected range of the **pretrained model**, which was originally trained on images normalized to the [-1,1] range. This transformation ensures consistency with the backbone's training distribution, facilitating more effective feature extraction and stable convergence during fine-tuning.

Furthermore, image tensors were reformatted from the conventional **height × width × channels (HWC)** format to **channels × height × width (CHW)** to comply with PyTorch's tensor input specifications.

Note: Data augmentation, although technically applied to the images at this stage, is detailed separately in **Section 2.3 (Data Augmentation)** to emphasize its methodological role in improving model generalization.

2.2.2 Mask Preprocessing

For the segmentation masks, preprocessing aimed to ensure binary consistency and alignment with the corresponding images. Initially, masks were loaded in grayscale mode and resized using nearest-neighbor interpolation to preserve categorical values without introducing interpolation artifacts.

Originally, the preprocessing pipeline included a step to scale mask pixel values from [0, 255] to [0, 1], followed by thresholding at 0.5:

$$M_{\text{binary}} = \begin{cases} 1 & \text{if } M > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

However, after manually inspecting all the samples from both the Massachusetts and DeepGlobe datasets, we observed that the masks were already binarized, containing only pixel values of 0 (background) and 1 (road). Therefore, the thresholding operation was redundant and could be safely omitted from the pipeline without affecting data integrity.

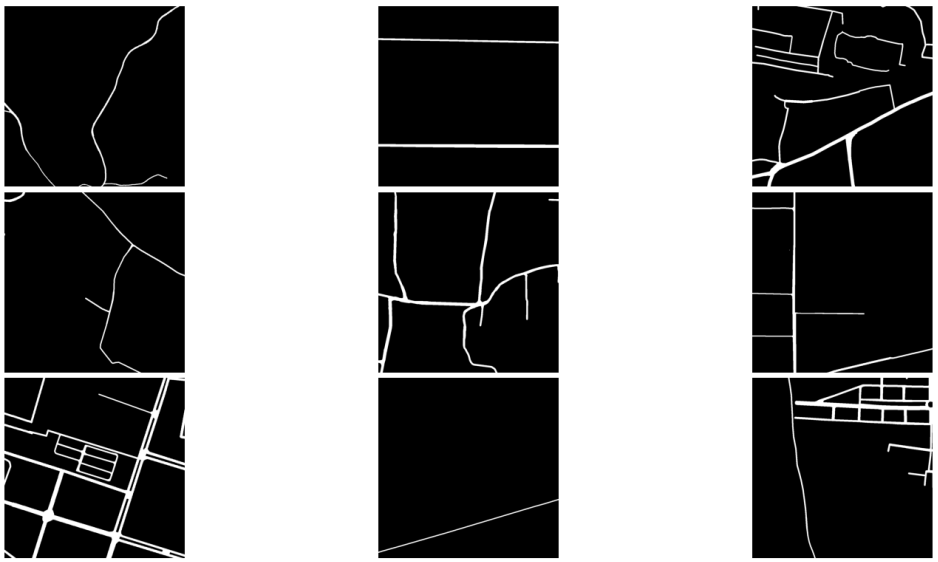

Figure 7 Some mask samples

The binary masks were subsequently expanded with an additional singleton channel dimension to match the expected input shape for our models:

$$M_{\text{final}} \in \mathbb{R}^{1 \times H \times W}$$

The binary nature of the masks directly supports the formulation of the segmentation task as a binary pixel-wise classification, predicting whether each pixel belongs to a road or background.

In summary, the image and mask processing pipeline ensured all inputs were standardized in terms of spatial resolution, value range, and tensor shape. The image normalization formula was adapted from prior work [3], proven effective for

similar segmentation tasks, while the mask pipeline benefited from the datasets' consistent labeling conventions, obviating the need for additional thresholding.

Additionally, the incorporation of data augmentation enhanced the model's ability to generalize to unseen images by exposing it to diverse variations during training.

2.3 Data Augmentation

To enhance the robustness and generalization capability of the model, a set of data augmentation techniques was incorporated into the training pipeline. These augmentations introduced variability in color, orientation, and geometry of the training images, effectively simulating diverse imaging conditions and reducing the risk of overfitting.

All augmentations were applied on-the-fly during data loading using OpenCV functions in the custom PyTorch Dataset class, ensuring that each training epoch saw slightly different variations of the same image.

The following augmentations were implemented:

1. **Hue, Saturation, and Value Shift:** Random changes to color components simulate varying lighting conditions, seasonal changes, or atmospheric effects.

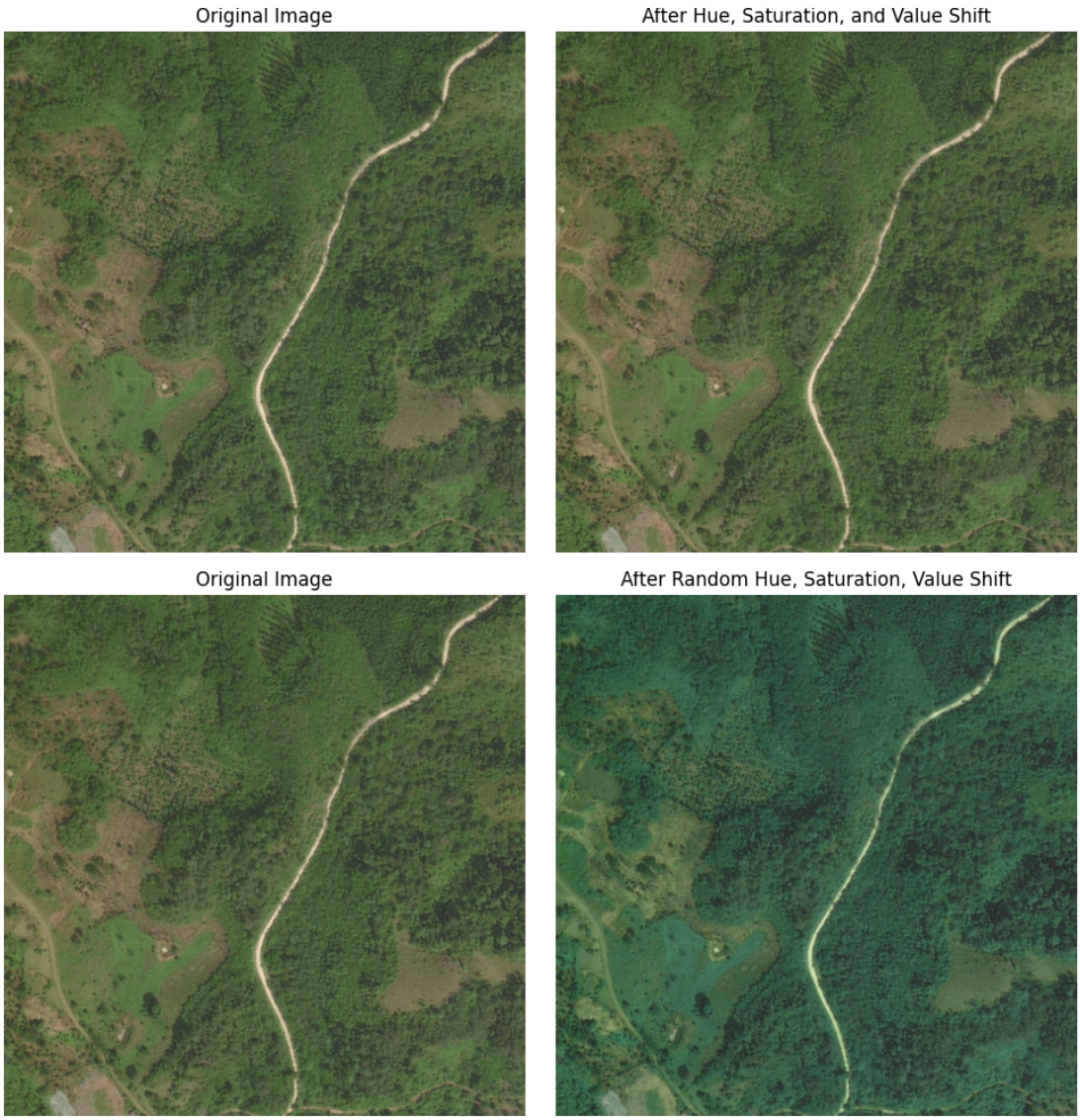

Figure 8 Random Hue, Saturation, Value Shift

2. **Random Shift, Scale, and Rotate:** Geometric transformations such as translation, scaling, and minor rotations help the model generalize to positional variations in the road network.

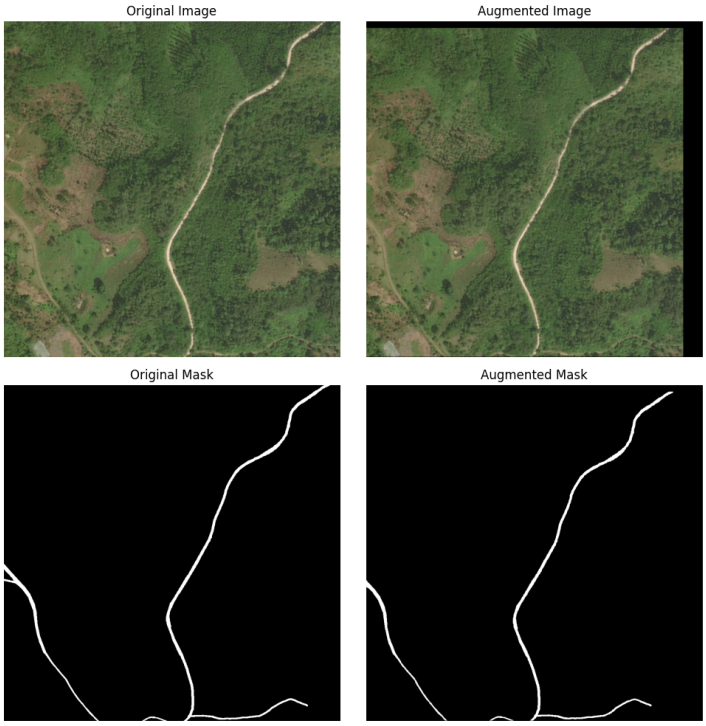

Figure 9 Random Shift, Scale, Rotate

3. **Random Horizontal and Vertical Flips:** These augmentations introduce invariance to image orientation, accounting for different map orientations in satellite imagery.

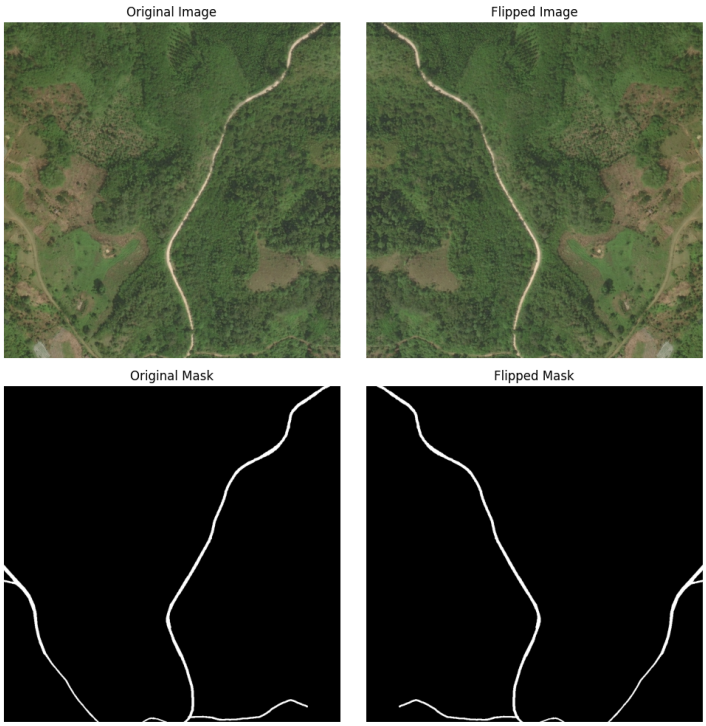

Figure 10 Random Horizontal Flip

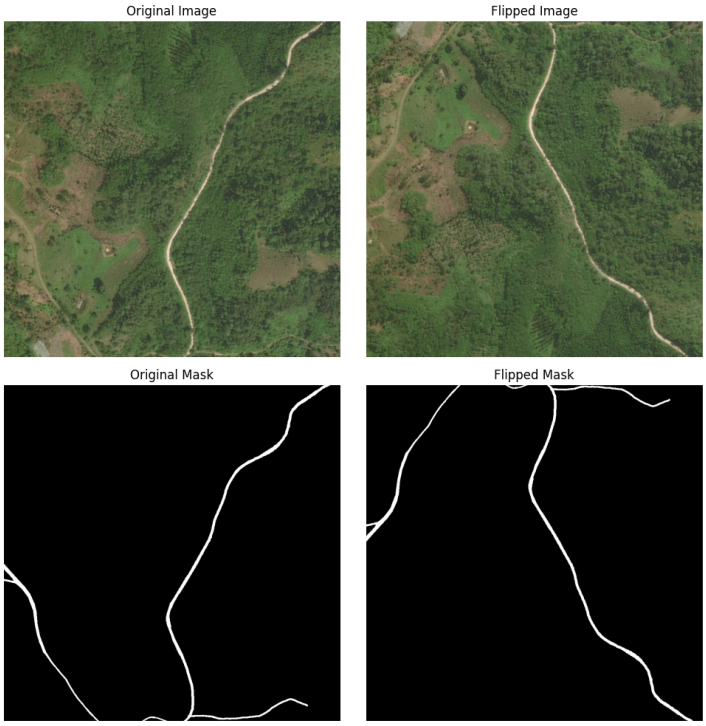

Figure 11 Random Vertical Flip

4. **Random 90° Rotations:** Further increases orientation robustness by randomizing the directional layout of roads.

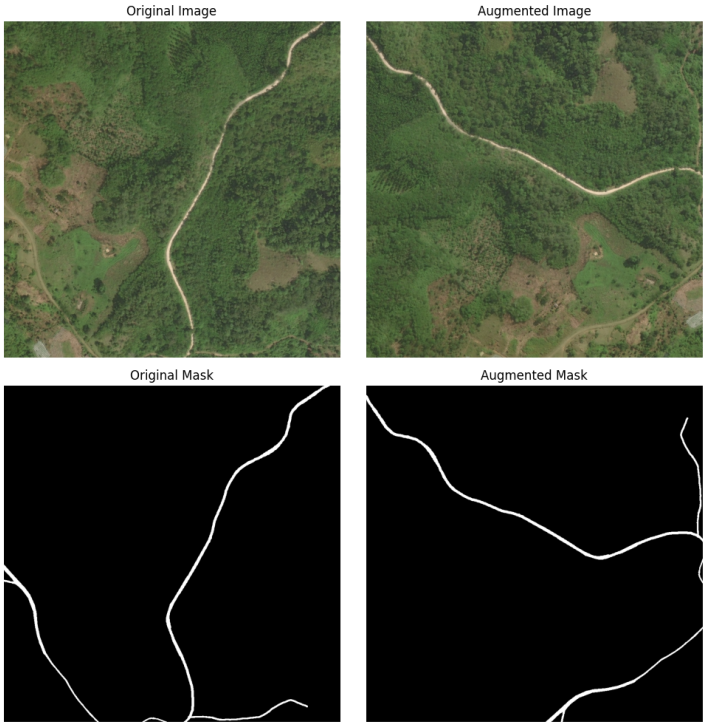

Figure 12 Random 90° Rotation

All augmentations were applied to both the image and its corresponding mask si-

multaneously to maintain pixel-wise correspondence between input and label, except for the Random Hue, Saturation, Value Shift, which was only applied on the image.

By combining photometric and geometric augmentations, the dataset was effectively expanded, providing the model with a more diverse and challenging training environment.

2.4 Data Loading Pipeline

After preprocessing, the dataset was integrated into a custom PyTorch Dataset class to streamline loading, preprocessing, and augmentation within a unified framework. Each data sample returned by the Dataset class consists of a normalized input image tensor and a corresponding binary mask tensor, both formatted as PyTorch tensors ready for model input.

A PyTorch DataLoader was employed to efficiently handle batching, shuffling, and parallel data loading across multiple workers. The data loading pipeline was configured as follows:

- **Batch size:** 4 samples per batch
- **Shuffling:** Enabled for training set, disabled for validation and test sets
- **Number of workers:** 4 parallel workers to accelerate loading
- **Split ratio:** 70% training, 15% validation, 15% testing

Data augmentations (described in Section 2.3) were applied on-the-fly during data loading to ensure each epoch receives diverse input samples.

This modular data pipeline ensured seamless integration with the training loop, optimized GPU utilization, and supported real-time data augmentation during training.

CHAPTER III. EVALUATION METRICS

In this project, the performance of the segmentation model was evaluated using two primary metrics: the Dice coefficient and the Intersection over Union (IoU). Both metrics are derived from the concept of overlap between the predicted segmentation masks and the ground truth masks, and are based on pixel-level comparisons. Before introducing the formulas, it is important to clarify the underlying confusion matrix concepts as applied to image segmentation.

3.1 Confusion Matrix for Image Segmentation

In image segmentation tasks, the performance at each pixel is classified into four possible outcomes, similar to classification tasks but applied at the pixel level:

- **True Positive (TP):** a pixel correctly predicted as belonging to the object (foreground).
- **False Positive (FP):** a pixel incorrectly predicted as belonging to the object, while it actually belongs to the background.
- **False Negative (FN):** a pixel incorrectly predicted as background, while it actually belongs to the object.
- **True Negative (TN):** a pixel correctly predicted as background.

	Positive (1)	Negative (0)
Postive (1)	True Positive	False Positive
Negative (0)	False Negative	True Negative

Table 1 Confusion matrix for binary classification

In segmentation tasks, the focus is usually on the object (foreground) class, so most evaluation metrics are derived from **TP**, **FP**, and **FN**. The values of **TP**, **FP**, and **FN** are accumulated across all pixels in the predicted mask and ground truth mask.

These pixel-level confusion matrix components are then used to calculate overlap-based metrics such as the Dice coefficient and Intersection over Union.

In addition to Dice and IoU, three fundamental classification metrics are also commonly used to further evaluate segmentation quality, namely **Precision**, **Recall**,

and **F1-Score**. These metrics provide a more complete picture of model behavior, especially in use cases where minimizing either false positives or false negatives is more critical.

3.2 Dice Coefficient

The Dice coefficient (also referred to as the Sørensen–Dice index) is a statistic that gauges the similarity between two sets. In the context of image segmentation, it evaluates how well the predicted mask aligns with the ground truth. Mathematically, it is defined as:

$$\text{Dice} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}$$

or equivalently:

$$\text{Dice} = \frac{2 \times |\text{Prediction} \cap \text{GroundTruth}|}{|\text{Prediction}| + |\text{GroundTruth}|}$$

where:

- $|\text{Prediction} \cap \text{GroundTruth}|$ is the number of pixels correctly predicted as object (i.e., True Positives),
- $|\text{Prediction}|$ is the total number of pixels predicted as object (i.e., **TP + FP**),
- $|\text{GroundTruth}|$ is the total number of object pixels in the ground truth (i.e., **TP + FN**).

The Dice coefficient ranges from 0 (no overlap) to 1 (perfect overlap), and is particularly robust in handling class imbalance in segmentation tasks.

3.3 Intersection over Union (IoU)

The Intersection over Union (IoU) metric measures the proportion of the overlap between the predicted and ground truth masks relative to their union. It is computed as:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$$

or equivalently:

$$\text{IoU} = \frac{|\text{Prediction} \cap \text{GroundTruth}|}{|\text{Prediction} \cup \text{GroundTruth}|}$$

with:

- $|\text{Prediction} \cap \text{GroundTruth}|$ corresponding to True Positives (**TP**),
- $|\text{Prediction} \cup \text{GroundTruth}|$ equivalent to **TP + FP + FN**, representing all pixels that are either predicted or actually belong to the object.

Similar to the Dice coefficient, IoU values range from 0 to 1, with higher values indicating better segmentation quality. While Dice and IoU are mathematically related, IoU tends to be a stricter measure of overlap and is sensitive to small discrepancies between predicted and actual masks.

Both Dice and IoU were computed on the validation dataset after each training epoch to monitor the segmentation quality. Higher values of these metrics indicate better overlap between the predicted and true segmentation masks.

Note: While both metrics were tracked during validation, model training used the Dice coefficient as the primary criterion for performance monitoring. This allowed us to employ early stopping based on improvements in Dice loss, prioritizing overlap quality in our stopping decision.

3.4 Confusion Matrix-based Metrics: Precision, Recall, and F1 Score

In addition to segmentation-specific metrics such as the Dice coefficient and Intersection over Union (IoU), model performance can also be evaluated using traditional metrics derived from the confusion matrix, including **Precision**, **Recall**, and the **F1 Score**. These metrics provide deeper insights into how well the model performs in different aspects, such as its ability to correctly identify object pixels or avoid false detections. They are particularly useful in scenarios with class imbalance, where background pixels dominate and objects occupy only a small portion of the image.

1. **Precision**

   Precision quantifies the accuracy of the model in predicting object regions. It is defined as the ratio of correctly predicted object pixels (True Positives – TP) to the total number of pixels predicted as objects (TP + FP):

$$\text{Precision} = \frac{TP}{TP + FP}$$

A high precision indicates that the model produces few **false positives**, meaning it rarely misclassifies background pixels as objects.

2. **Recall**

   Recall, also known as sensitivity, measures the model's ability to detect all actual object pixels. It is calculated as the ratio of correctly predicted object pixels (TP) to the total number of ground truth object pixels (TP + FN):

$$\text{Recall} = \frac{TP}{TP + FN}$$

   A high recall score suggests that the model **misses few object regions**, ensuring better coverage of the object areas in the image.

3. **F1-Score**

   The F1 Score is the harmonic mean of Precision and Recall, offering a balanced measure that considers both correctness and completeness:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

   The F1 Score only achieves high values when both Precision and Recall are high. It is particularly useful when dealing with **imbalanced datasets**, where a balance between identifying all objects and avoiding false positives is critical.

Like the Dice coefficient and IoU, the values of Precision, Recall, and F1 Score range from 0 to 1, where 1 indicates perfect segmentation performance, and 0 indicates no overlap between predictions and ground truth. These confusion matrix-based metrics played an important role in diagnosing model behavior, for instance:

- **Low Precision** $\rightarrow$ The model frequently misclassifies background as object (over-segmentation).
- **Low Recall** $\rightarrow$ The model fails to detect many object pixels (under-segmentation).
- **Low F1 Score** $\rightarrow$ indicates an imbalance between Precision and Recall.

Tracking these metrics during validation helped ensure the model not only achieved high overlap with ground truth but also maintained a balanced detection strategy, which is crucial for generalization to unseen data.

CHAPTER IV. METHODOLOGY

4.1 Introduction to Semantic Segmentation

Semantic segmentation is a fundamental task in computer vision, aimed at assigning a semantic label (such as "road", "building", or "background") to every pixel in an image. Unlike image classification, which predicts a single label per image, or object detection, which predicts bounding boxes for objects, semantic segmentation provides **dense, pixel-level** understanding of the scene.

In recent years, semantic segmentation has gained significant attention due to its applications in various domains, such as:

- Autonomous driving (road and lane detection)
- Medical imaging (organ or tumor segmentation)
- Remote sensing (building or road extraction from satellite imagery)
- Augmented reality and robotics.

Formally, given an input image $I \in \mathbb{R}^{H \times W \times 3}$, semantic segmentation aims to produce a mask $M \in \{0, 1, ..., C-1\}^{H \times W}$, where each pixel is classified into one of C semantic categories.

With the advent of deep learning, convolutional neural networks (CNNs) have become the dominant approach for semantic segmentation, replacing earlier methods based on hand-crafted features or probabilistic graphical models. Architectures such as FCN (Fully Convolutional Network), U-Net, DeepLab, and D-LinkNet have been proposed to tackle the challenges of this task, including:

- Preserving fine details despite downsampling.
- Capturing global context for disambiguating similar regions.
- Efficiently recovering spatial resolution through upsampling.

In this project, we focus on the specific task of road extraction from high-resolution satellite imagery, a challenging scenario where roads may be occluded, thin, or similar in color to their surroundings. The models chosen for this task are D-LinkNet (which would be built from scratch) and the fine-tuned model SegFormer. All of them are encoder-decoder architectures specifically designed for segmenting elongated structures such as roads.

4.2 DLinkNet Model

D-LinkNet, introduced by Zhou et al. (2018), is a deep convolutional neural network tailored for road extraction in high-resolution aerial and satellite images. It builds upon the encoder-decoder paradigm, combining the strengths of a pretrained ResNet encoder, a dilated convolutional center block, and a lightweight decoder with skip connections.

The key motivation behind D-LinkNet is to address the following challenges in road segmentation:

1. Roads are long, thin, and may span across large areas $\rightarrow$ requires a large receptive field.
2. Roads have subtle visual differences from surrounding terrain $\rightarrow$ needs multi-scale context.
3. Precise boundary localization is critical $\rightarrow$ skip connections help preserve fine details.

The architecture integrates:

- A **ResNet34** backbone pretrained on ImageNet as the encoder for robust feature extraction.
- A **Dilated Convolution Block (DBlock)** at the bottleneck to expand the receptive field without losing resolution.
- A **Decoder** consisting of upsampling and convolutional layers to progressively reconstruct the segmentation map.
- **Skip connections** to merge low-level and high-level features.

D-LinkNet achieves a balance between depth (thanks to ResNet34), context (via DBlock), and spatial precision (via skip connections), making it suitable for road extraction tasks.

In this section, we provide a detailed breakdown of each component of the D-LinkNet architecture, including its basic building blocks, encoder, center block, decoder, skip connections, and final output layers.

4.2.1 Basic Blocks

Before delving into the detailed architecture of DLinkNet, it is essential to understand the fundamental building blocks that constitute the model. DLinkNet, like many convolutional neural networks, leverages several core types of layers to process image data. Each layer type plays a specific role in feature extraction, non-linear transformation, normalization, downsampling, or upsampling.

1. **Convolutional Layer (Conv2d)**

   A **convolutional layer (Conv2d)** is the cornerstone of deep learning models for image tasks. It applies a set of **kernels (learnable filters, sliding windows)** on the input images or feature maps, sliding the filters spatially across the image to compute feature activations.

   Mathematically, a convolution operation is defined as:

$$\text{output}(i,j) = \sum_{m,n} \text{input}(i+m, j+n) \cdot \text{kernel}(m,n) + \text{bias}$$

   Where:

   - $input$ = input feature map
   - $kernel$ = convolution filter
   - $output$ = output feature map.

   In DLinkNet, `Conv2d` layers are used across the encoder, decoder, and center block to:

   - Extract spatial features (edges, textures).
   - Increase or transform feature depth.
   - Maintain spatial locality (each output pixel depends only on nearby input pixels).

   The main hyperparameters are:

   - **Kernel size:** usually 3x3
   - **Stride:** usually 1
   - **Padding:** adds zero-value pixels around the input image to preserve the input size

The basic operation of Convolution Layer is illustrated as below:

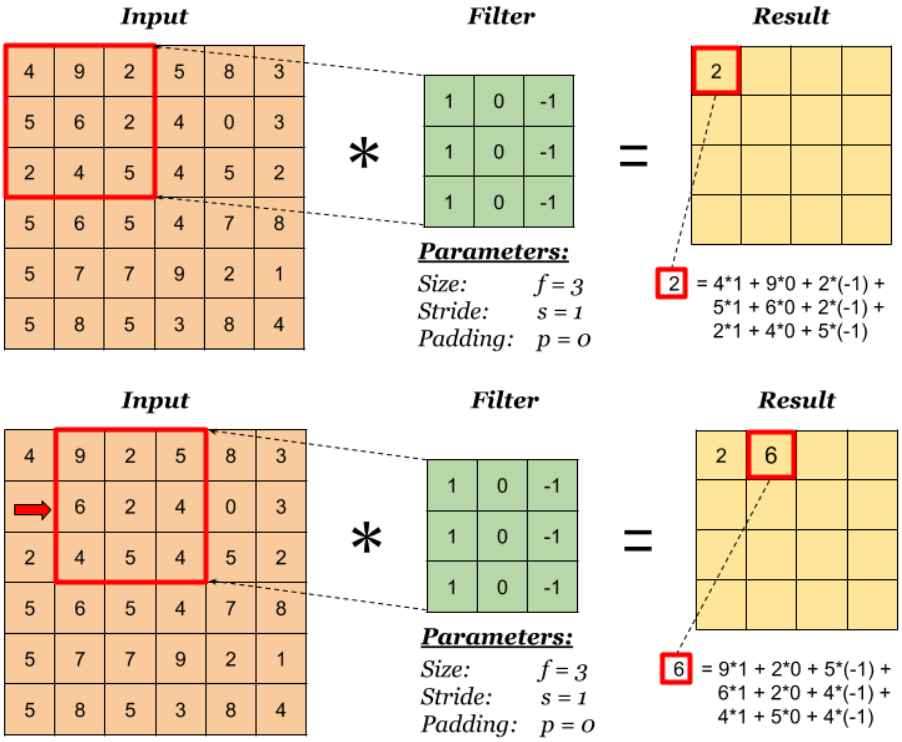

Figure 13 An example for Basic Convolution Operation

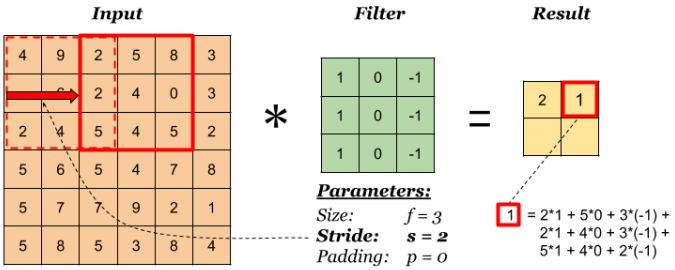

Figure 14 Convolution Stride

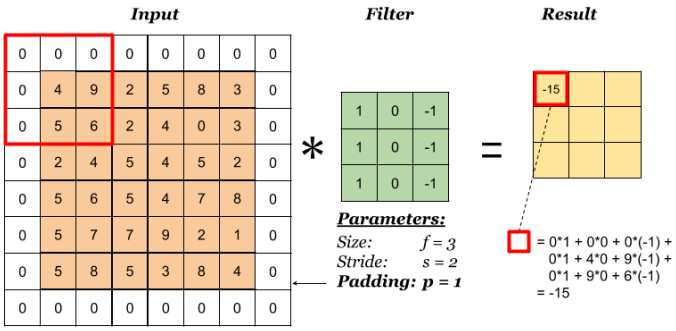

Figure 15 Convolution Padding

2. **Rectified Linear Unit (ReLU)**

   The **ReLU** activation function introduces **non-linearity** into the network by applying the function:

$$\text{ReLU}(x) = \max(0, x)$$

   It replaces all negative values with zero while keeping positive values unchanged.

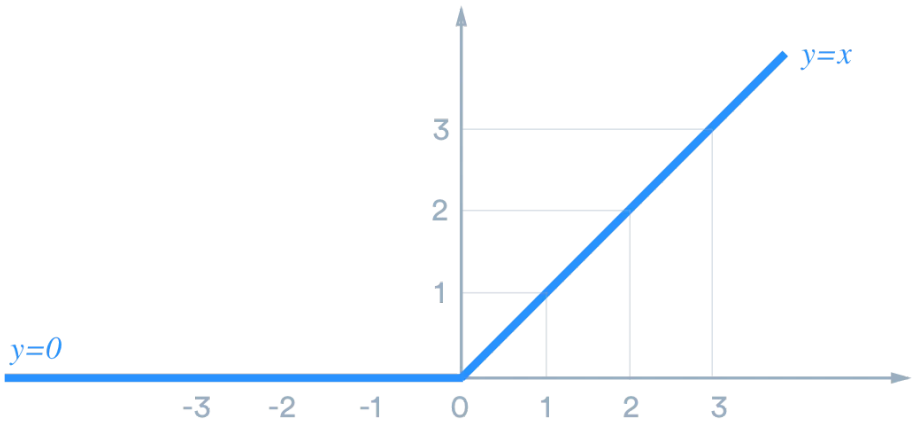

Figure 16 ReLU Activation Function

   In DLinkNet:

   - ReLU is applied after most convolutional layers to enable the model to learn complex, non-linear patterns.
   - Helps alleviate the vanishing gradient problem.

   Key properties:

   - Simple computation.
   - Prevents saturation (compared to sigmoid, tanh).
   - Introduces sparsity in activations, as neurons may output zero for negative inputs.

3. **Batch Normalization (BatchNorm2d)**

   Batch normalization normalizes the output of a layer by subtracting the batch mean and dividing by the batch standard deviation, followed by learnable scaling and shifting.

Given a mini-batch $B = \{x_1, x_2, ..., x_m\}$ of activations (per channel), batch normalization is computed as:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{(mean)}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad \text{(variance)}$$

Each activation x_i is then normalized:

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

where: ε is a small constant to avoid division by zero.

Finally, two learnable parameters γ and β allow the model to scale and shift the normalized output:

$$y_i = \gamma\hat{x_i} + \beta$$

Role in DLinkNet:

- Applied after convolution, before ReLU.
- Helps stabilize training by reducing **internal covariate shift.**
- Allows higher learning rates without risk of divergence.
- Improves convergence speed.

4. **Max Pooling (MaxPool2d)**

   Max pooling is a downsampling operation that reduces the spatial resolution of a feature map by taking the **maximum value** within each local region (window). For example, a 2×2 max pool with stride 2 halves the spatial dimensions.

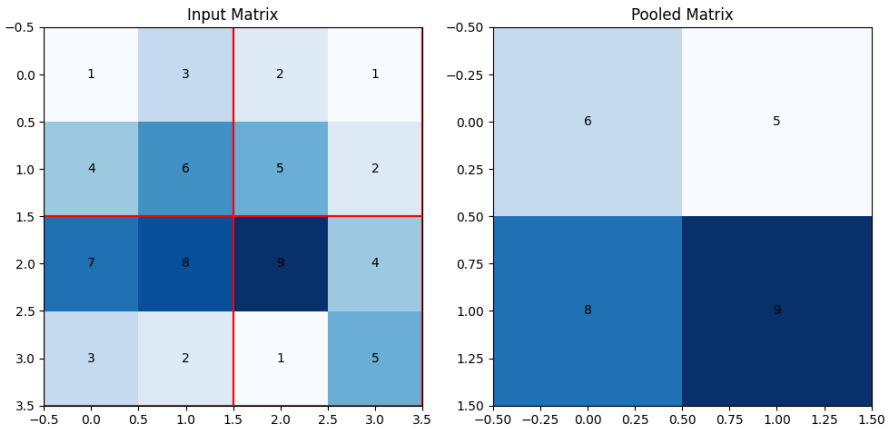

Figure 17 Max Pooling Operation

In DLinkNet:

- Used in the initial encoder layers (ResNet34's `maxpool`).
- Purpose:
  - Reduce computation and memory.
  - Increase receptive field.
  - Retain dominant features (edges, textures).

5. **Transposed Convolution (ConvTranspose2d & ConvTranspose3d)**

   Transposed convolution (also called **deconvolution** or **upconvolution**) performs the inverse operation of a regular convolution: it increases the spatial resolution.

   A normal convolution computes **dot products** between the kernel and overlapping regions of the input, producing a smaller output.

   A transposed convolution starts with a small input and **spreads it into a larger output** by inserting zeros (or empty spaces) between pixels and then applying the kernel, effectively "filling in" the larger output.

   An intuitive way to think about it:

   - **Normal convolution:** sliding window $\rightarrow$ compress info.
   - **Transposed convolution:** reverse sliding $\rightarrow$ expand info.

   A concrete example to explain the operation of a transposed convolution: Suppose we have a grayscale image of size 2×2, and we want to upsample it using

a transposed convolutional layer with a kernel size of 2×2, a stride of 1, and zero padding (or no padding). The operation is depicted as below:

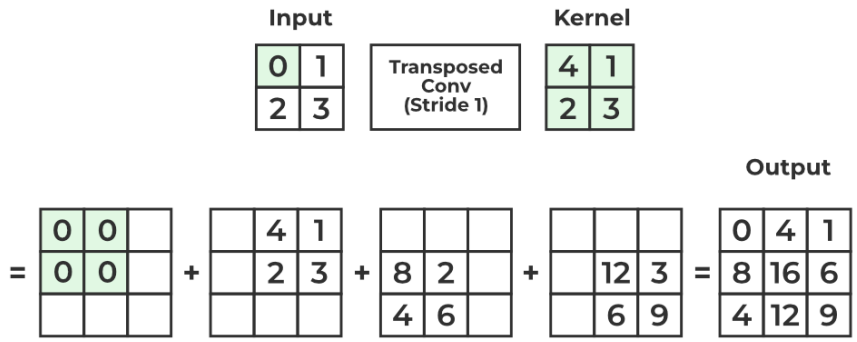

Figure 18 Transpose Convolution Operation

ConvTranspose2d

A transposed 2D convolution increases width and height:

- Used in DLinkNet's **final decoder layers** to upsample the feature map towards the original input resolution.
- Allows the model to learn **learnable upsampling kernels**, in contrast to fixed upsampling (e.g., bilinear).

ConvTranspose3d

In DLinkNet's decoder block, a **ConvTranspose3d** is applied on tensors with an added singleton dimension (shape: B, C, 1, H, W) to perform 2D transposed convolution while technically using 3D operator:

$$
\begin{aligned}
x &= x.\text{unsqueeze}(2) \qquad (B, C, 1, H, W) \\
x &= \text{ConvTranspose3d}(x) \\
x &= x.\text{squeeze}(2)
\end{aligned}
$$

Reason:

- This trick allows the reuse of existing 3D transpose logic for 2D upsampling in PyTorch.
- Behaves similarly to ConvTranspose2d on the spatial dimensions.

4.2.2 Encoder Module (ResNet34)

The encoder of DLinkNet is built upon the **ResNet34** architecture, which serves as a powerful feature extractor. ResNet34 is a deep residual network consisting of 34 convolutional layers arranged into multiple blocks with **skip connections** (residual connections). These connections allow the network to bypass some layers by adding the input of a block directly to its output, effectively mitigating the problem of **vanishing gradients** and enabling the training of deeper networks.

The encoder in our DLinkNet model leverages the pretrained ResNet34 from ImageNet. The architecture includes the following key components:

Initial Convolution and Pooling:

- A 7x7 convolution with stride 2 and padding 3.
- Followed by a Batch Normalization and a ReLU activation.
- Then a 3×3 max pooling layer with stride 2, padding 1.

Four residual blocks:

- **Layer 1:** Contains 3 basic residual blocks (64 filters).
- **Layer 2:** Contains 4 residual blocks (128 filters), with downsampling at the first block.
- **Layer 3:** Contains 6 residual blocks (256 filters), with downsampling.
- **Layer 4:** Contains 3 residual blocks (512 filters), with downsampling.

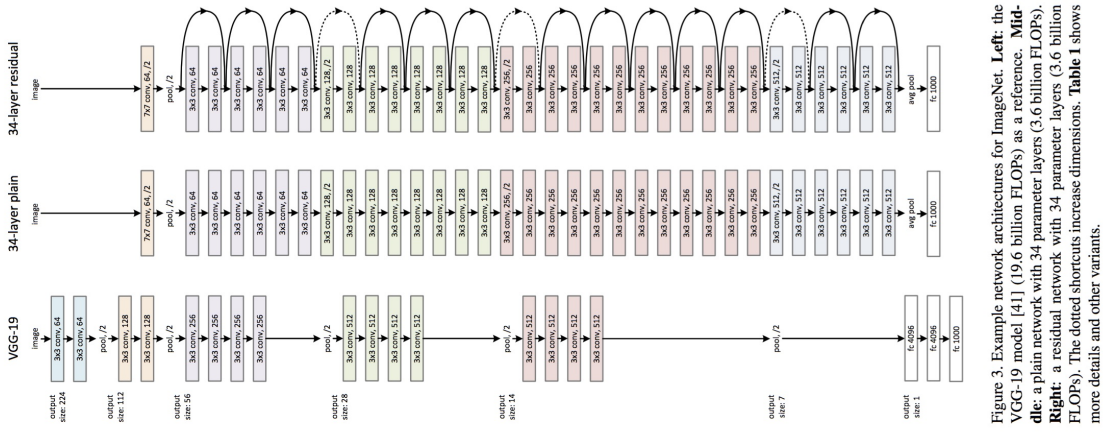

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Figure 19 Resnet34 (image from http://euler.stat.yale.edu)

At each stage, the spatial resolution of the feature map is reduced while the number of feature channels increases. This progressive downsampling allows the network to capture high-level semantic information.

The encoder outputs multi-scale feature maps at different depths, which are later utilized by the decoder via **skip connections** to recover spatial details.

4.2.3 Center Block (Dilated Convolution Block – DBlock)

At the bottleneck of DLinkNet lies a specially designed module known as the **Dilated Convolution Block (DBlock)**. This block plays a critical role in **expanding the receptive field** without reducing the resolution of the feature map.

Dilated (or atrous) convolutions introduce "holes" into the convolution kernel by skipping certain positions, effectively allowing the kernel to cover a larger area without increasing the number of parameters. This is particularly useful in segmentation tasks where the model needs to incorporate broader contextual information while preserving spatial resolution.

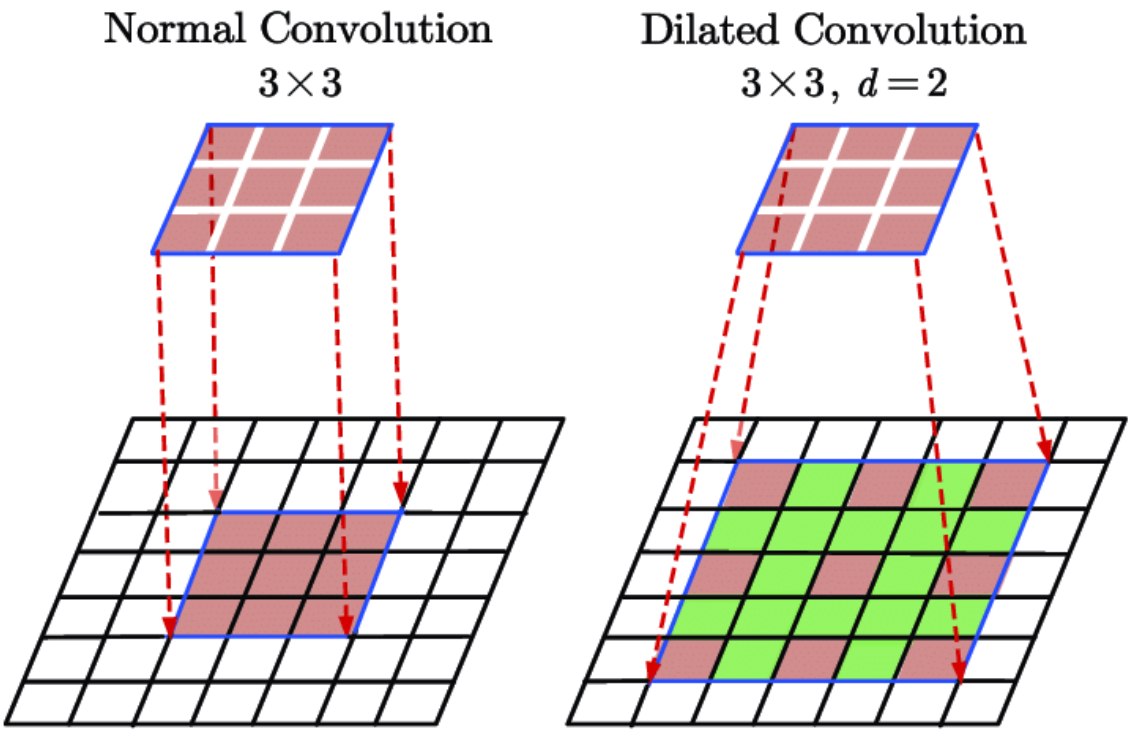

Figure 20 Dilated Convolution

In DLinkNet, the DBlock consists of four sequential convolution layers with increasing dilation rates: $1 \rightarrow 2 \rightarrow 4 \rightarrow 8$

Each layer applies 3x3 convolution, padding equal to the dilation rate to maintain output size, and ReLU activation.

Mathematically, for an input feature map x, the output y of DBlock can be expressed as:

$$\begin{aligned} y = x &+ \text{ReLU}(\text{Conv}_{d=1}(x)) \\ &+ \text{ReLU}(\text{Conv}_{d=2}(x_1)) \\ &+ \text{ReLU}(\text{Conv}_{d=4}(x_2)) \\ &+ \text{ReLU}(\text{Conv}_{d=8}(x_3)) \end{aligned} \tag{1}$$

where x_1, x_2, x_3 are the intermediate outputs after each dilated convolution.

The addition of input x implements a residual connection, helping gradient flow and retaining low-level features.

4.2.4 Decoder Module (DecoderBlock)

The decoder module in DLinkNet is designed to progressively upsample the feature maps and recover the original image resolution while combining high-level and low-level features through skip connections.

Each DecoderBlock follows a structured process:

1. **1×1 convolution** to reduce channel dimension.
2. **ConvTranspose3d** performs a 3D transposed convolution on an expanded dimension (a workaround simulating 2D transposed convolution by temporarily adding a singleton dimension), which upsample the spatial resolution by a factor of 2.
3. **Batch Normalization + ReLU activation**.
4. **1×1 convolution** to restore the desired number of output channels.
5. **Batch Normalization + ReLU**.

The use of ConvTranspose3d is a trick to achieve 2D transposed convolution with specific kernel behavior, due to framework constraints.

Formally, each DecoderBlock transforms a feature map of shape $C_{in} \times H \times W$ to an output $C_{out} \times 2H \times 2W$, allowing the decoder to double the spatial resolution at each stage.

Each decoder block also adds skip connections from the corresponding encoder feature map, improving localization by blending semantic and spatial information.

4.2.5 Final Layers

After passing through the decoder, the feature map is further refined by a sequence of final layers aiming to produce the segmentation mask at the original image resolution.

The final layers include:

1. **ConvTranspose2d (4×4 kernel, stride 2):** Upsamples the output from the last decoder block by a factor of 2.
2. **ReLU activation**.
3. **Conv2d (3×3 kernel):** Refines the upsampled features.
4. **ReLU activation**.
5. **Conv2d (3×3 kernel, output channels = number of classes):** Generates the final logit map for segmentation.
6. **Sigmoid activation:** Converts logits to probabilities in range [0,1] for binary segmentation.

This final step ensures that the output has the same height and width as the input image, enabling pixel-wise classification.

The sigmoid activation is crucial for the binary segmentation task, as it models the probability of each pixel belonging to the road class.

4.2.6 Skip Connections in DLinkNet

Skip connections, also known as **shortcut connections**, play a crucial role in the architecture of DLinkNet. Their primary function is to **bridge the encoder and decoder**, allowing the model to leverage both high-level semantic features and low-level spatial details.

In the context of semantic segmentation, skip connections are essential. This is because, as the network goes deeper, spatial information (such as edges, fine boundaries, small structures) tends to get lost due to successive downsampling operations (pooling, strided convolution). Without skip connections, the decoder would have to reconstruct fine-grained details solely from coarse, abstract features. This is challenging and often leads to blurry or even inaccurate segmentation outputs.

DLinkNet addresses this by **directly adding the output of encoder layers to the corresponding decoder layers of the same spatial scale**. Specifically:

$$d_4 = DecoderBlock_4(center) + e_3$$
$$d_3 = DecoderBlock_4(d4) + e_2$$
$$d_2 = DecoderBlock_4(d3) + e_1$$

Where:

- e_1, e_2, e_3 are the outputs from encoder layers (after layer1, layer2, layer3 of ResNet34).
- d_4, d_3, d_2 are outputs of decoder blocks.
- $center$ is the output of the dilated convolution block.

Each skip connection performs an **element-wise addition** of the decoder output and the encoder feature map at the same resolution. This operation is possible because the decoder uses upsampling to match the resolution of the encoder outputs at each stage.

By integrating skip connections, DLinkNet inherits the advantages of both **global semantic context** (from deeper encoder layers) and **precise localization cues** (from earlier encoder layers).

The inclusion of skip connections in DLinkNet is motivated by findings in prior works, which have shown that direct connections between encoder and decoder layers at matching resolutions enable the model to combine high-level semantic features with low-level spatial details.

Specifically, the reported benefits in literature include:

- **Improved preservation of fine spatial details**, such as thin structures (e.g., roads or vessels), by providing the decoder access to high-resolution features from the encoder.
- **Facilitated gradient flow** during backpropagation, reducing the risk of vanishing gradients and improving convergence speed and stability.
- **Reduced information bottleneck** between encoder and decoder by directly passing intermediate feature maps.

This architectural design has been empirically validated in the original D-LinkNet and U-Net studies to yield sharper segmentation boundaries and better reconstruction of small objects, especially in tasks involving long, narrow structures.

In this project, we adopted the skip connection strategy as implemented in the open-source D-LinkNet repository [3], without ablation experiments to remove skip connections. Therefore, the expected benefits are aligned with the findings reported in existing literature (Zhou et al., 2018 [4]).

Explanation in simpler terms

We can think of skip connections as "reminders" from the earlier layers, telling the decoder: “Hey, here’s what the original image looked like at this scale—don’t forget these details while you’re reconstructing!” Without these connections, the decoder might struggle to reconstruct the fine structures lost during downsampling.

4.2.7 Summary Table of Architecture

The table below summarizes the architecture of DLinkNet34 implemented in this study, showing the output size and number of channels after each key stage of the network.

Stage	Layer	Output Shape (C × H × W)
Input	Input Image	3 × 1024 × 1024
Encoder Block 0	Conv7×7 + BN + ReLU + MaxPool	64 × 256 × 256
Encoder Block 1	ResNet34 Layer1	64 × 256 × 256
Encoder Block 2	ResNet34 Layer2	128 × 128 × 128
Encoder Block 3	ResNet34 Layer3	256 × 64 × 64
Encoder Block 4	ResNet34 Layer4	512 × 32 × 32
Center Block	DBlock (dilated conv)	512 × 32 × 32
Decoder Block 4	Decoder4 + Skip(e3)	256 × 64 × 64
Decoder Block 3	Decoder3 + Skip(e2)	128 × 128 × 128
Decoder Block 2	Decoder2 + Skip(e1)	64 × 256 × 256
Decoder Block 1	Decoder1	64 × 256 × 256
Final Layers	TransConv + Conv + Conv + Sigmoid	1 × 1024 × 1024

Table 2 Network Architecture Overview

As shown in the table, the model follows an encoder-center-decoder structure, progressively downsampling the input and then upsampling it back to the original resolution. Skip connections are applied from Encoder Block 1–3 to Decoder Block 2–4 to retain spatial details.

4.3 Pretrained Models

4.3.1 Selection of SegFormer-B1

In addition to our primary architecture, D-LinkNet, we also consider **SegFormer-B1**, a pretrained transformer-based semantic segmentation model. SegFormer, introduced by NVIDIA, represents a shift from convolutional networks to transformer-based architectures for dense prediction tasks. Its design is centered around two main components:

- A **Mix Vision Transformer (MiT)** encoder that extracts multi-scale features hierarchically via progressive patch merging.
- A **lightweight decoder** composed entirely of multilayer perceptrons (MLPs), avoiding the use of transposed convolutions or complex upsampling paths.

This architecture enables **global context modeling** with fewer parameters and without the need for positional encodings or heavy decoders. Unlike conventional CNN-based models, SegFormer can effectively capture long-range spatial dependencies, which are crucial for road segmentation in satellite imagery, where road segments may span large areas of the image.

Among the available SegFormer variants (B0 through B5), we selected **SegFormer-B1** due to its **balanced trade-off** between accuracy and computational complexity. Compared to B0, B1 provides better feature representation while still maintaining reasonable training and inference efficiency. Heavier variants (B2–B5), although more powerful, introduce substantial computational overhead, which is less desirable for our experiment-focused setup.

We utilized the official pretrained checkpoint, which was trained on the **ADE20K** dataset containing 150 semantic classes. This checkpoint is publicly available via HuggingFace's `transformers` library and provides a robust initialization point for fine-tuning on downstream tasks.

The model expects input images to be **3-channel RGB**, **resized to 512×512 pixels**, and **normalized to the range** [-1,1]. This normalization aligns the input distribution with the training distribution of the pretrained encoder and contributes to

stable convergence during fine-tuning.

4.3.2 Fine-tuning Strategy and Integration

To adapt SegFormer-B1 for our binary road segmentation task, several architectural and processing modifications were made.

First, the pretrained decoder head originally outputs **150-class segmentation maps**. We modify this when loading the model by setting the `num_labels` parameter to `1` and enabling `ignore_mismatched_sizes=True`. This allows the framework to automatically **replace the original classification head** with a single-channel convolution layer suitable for binary prediction.

Second, due to the hierarchical nature of the MiT encoder, the model outputs logits with **lower spatial resolution** than the input. In practice, for a 512×512 input, SegFormer-B1 typically produces an output of size 128×128. To ensure proper alignment with the ground-truth masks and to allow loss computation, we apply **bilinear interpolation** to upscale the logits back to the original size before applying the activation function and loss. Following this, we compute the segmentation loss using the **same loss function** as in D-LinkNet—**Binary Cross Entropy (BCE) combined with Dice loss**. This design ensures that both models are trained under consistent criteria and can be fairly compared.

The training data, augmentation pipeline, evaluation metrics, and other optimization strategies are kept identical to those used in D-LinkNet to isolate the effect of model architecture on performance.

4.4 Common Implementation Details

This section describes the key components and configurations involved in the training process of both our models, covering optimization strategies, learning rate scheduling, loss functions, early stopping criteria, evaluation procedures, software and hardware environment, and other implementation details.

4.4.1 Optimizer

An optimizer is a crucial algorithm that updates the neural network's parameters during training to minimize the loss function. The choice of optimizer affects both the speed of convergence and the quality of the learned model. In this project, we used AdamW, an improved version of the widely used Adam optimizer. To appreciate AdamW, we first explain Adam.

1. **Adam Optimizer: Adaptive Moment Estimation**

Adam (Adaptive Moment Estimation) is one of the most popular optimization algorithms for deep learning. It combines ideas from:

- **Momentum optimizer:** maintains a moving average of gradients (to smooth updates)
- **RMSProp optimizer:** maintains a moving average of squared gradients (to normalize updates)

At each iteration t, instead of using only the current gradient like SGD, Adam computes:

- m_t: exponentially decaying average of past gradients
- v_t: exponentially decaying average of past squared gradients

Formally:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

where:

- m_t: momentum at t
- v_t: smoothed gradient magnitude
- g_t: current gradient
- β_1, β_2: decay rates (typical values 0.9 and 0.999)

Because these averages are initialized at zero, they are biased towards zero at early steps $\rightarrow$ Adam introduces bias correction:

$$\hat{m}_t = \frac{m_t}{1-{\beta_1}^t}, \quad \hat{v}_t = \frac{v_t}{1-{\beta_2}^t}$$

The parameter update is:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

where:

- η: learning rate
- ε: small constant for numerical stability

2. **AdamW: Decoupled Weight Decay**

   In original Adam, adding L2 regularization is done **inside the gradient computation** (i.e., via loss term $\lambda ||\theta||^2$)

   However, this causes **interaction between weight decay and the adaptive updates**, leading to **ineffective regularization** (as shown by Loshchilov & Hutter, 2019[5]).

   AdamW fixes this by:

   - Keeping Adam's gradient update **as - is**
   - Applying weight decay **after** the gradient update as a separate step. The update becomes:

   $$\theta_{t+1} = \theta_t - \eta(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_t)$$

   This decouples weight decay from the adaptive gradient logic. Decoupling ensures:

   - Weight decay acts **purely as regularization**
   - No interference with moment estimates
   - More **effective generalization** (especially in deep models)

   In practice, AdamW improves performance in many tasks (vision, NLP) over Adam with L2 penalty.

   Road extraction involves detecting **small, thin structures**. Over-regularization could lead to underfitting, and under-regularization could lead to noisy predictions.

   AdamW gives:

   - Fast convergence (from Adam)
   - Better generalization control (via decoupled weight decay)
   - No need to tune learning rate decay schedule manually

   This made it ideal for training DLinkNet, where preserving edge details while avoiding overfitting is critical.

   In this project, we used AdamW with:

   - $\beta_1 = 0.9$, $\beta_2 = 0.999$

- weight decay $\lambda = 1e-4$

The optimizer contributed to stable training and improved final segmentation accuracy.

4.4.2 Learning Rate Strategy

The learning rate controls how much the model parameters change after each update. A high learning rate can lead to unstable training; too low leads to slow convergence. Instead of using a fixed learning rate, we used **ReduceLROnPlateau** scheduler, which dynamically lowers the learning rate if validation loss stops improving.

After each epoch, the scheduler checks the monitored metrics (validation loss). If no improvement is observed for **patience** epochs, the learning rate is reduced:

$$\text{new\_lr} = \text{current\_lr} \times \text{factor}$$

where:

- **patience:** number of epochs to wait before reducing
- **factor:** reduction multiplier (e.g., 0.5 halves the learning rate)

For example, suppose that: initial LR = 1e-4, patience = 3, factor = 0.5
$\rightarrow$ If validation loss doesn't improve after 3 epochs:

$$lr = (1e-4) \times 0.5 = 5e-5$$

$\rightarrow$ Wait another 3 epochs $\rightarrow$ if still no improvement:

$$lr = (5e-5) \times 0.5 = 2.5e-5$$

This gradual decay allows the optimizer to fine-tune around minima. We chose validation loss as the metric to monitor because loss is a direct signal of how well the model fits the data.

At the start of training, an initial learning rate of $1e-4$ was used, which is standard for Adam-based optimizers in segmentation tasks.

4.4.3 Loss Function

The choice of a loss function is critical because it defines how the model's predictions are penalized when they deviate from the ground truth. For semantic segmentation tasks, especially in the context of binary segmentation like road extraction,

a loss function needs to address both pixel-level accuracy and structural similarity between the predicted mask and the actual ground truth.

For both models, we used a combination of **Binary Cross Entropy (BCE)** and **Dice Loss**. This combination leverages the strengths of both local and global evaluation of predictions. While **BCE** focuses on pixel-wise accuracy (each pixel classified correctly), **Dice Loss** focuses on the overlap between predicted and ground truth regions.

The **Binary Cross-Entropy Loss** is commonly used for binary classification problems and is calculated independently for each pixel in the segmentation map. Mathematically, for a single pixel with true label $y \in \{0,1\}$ and predicted probability p, the BCE loss is defined as:

$$L_{BCE} = -[ylog(p) + (1-y)log(1-p)]$$

This loss measures how close the predicted probability is to the actual binary label. When summed over all pixels in an image, it penalizes every pixel that is classified incorrectly, regardless of its spatial relationship to other pixels. As such, BCE provides **fine-grained**, **pixel-level supervision**.

However, road segmentation presents a unique challenge: the class imbalance between road pixels and background pixels. Typically, the road occupies a small fraction of the image, leading to a situation where a naive model could achieve high accuracy by simply predicting "background" everywhere. To address this imbalance and encourage the model to focus on detecting road structures, we incorporate **Dice Loss**.

The Dice coefficient, from which Dice Loss is derived, measures the similarity between two sets. In the context of segmentation, it measures the overlap between the predicted mask p and the ground truth mask y:

$$Dice = \frac{2\sum(y \cdot p) + \varepsilon}{\sum y + \sum p + \varepsilon}$$

where ε is a small constant to avoid division by zero. The Dice Loss is then defined as:

$$L_{Dice} = 1 - Dice$$

Dice Loss is particularly useful in highly imbalanced settings because it directly optimizes for the **region-level overlap** rather than individual pixel errors. It encourages the model to predict complete road segments, even if they are thin or fragmented, which is crucial for road extraction tasks.

By combining BCE and Dice Loss, the total loss function can be written as:

$$L_{Total} = L_{BCE} + L_{Dice}$$

This combined loss allows the model to benefit from both the pixel-level precision of BCE and the region-level overlap optimization of Dice. The intuition is that BCE ensures that each pixel is treated carefully, while Dice Loss encourages the model to capture the overall shape and connectivity of road networks.

During training, this combination helped mitigate false negatives (missing small road segments) while maintaining high accuracy in larger regions. It effectively balances the trade-off between **local correctness and global structure preservation**, leading to better segmentation maps both visually and quantitatively.

In implementation, the loss function operates directly on the predicted probability maps output by the model (after sigmoid activation) and the binary ground truth masks. It plays a central role in guiding the optimizer towards parameters that minimize both individual pixel errors and overall mismatch in road area coverage.

In summary, the combination of Binary Cross Entropy and Dice Loss provides a robust learning signal for road segmentation, addressing the inherent class imbalance and structural complexity of the problem. Its integration into the training loop is pivotal for achieving accurate, detailed, and topologically sound segmentation results.

4.4.4 Early Stopping

During the training of deep neural networks, one of the key challenges is preventing overfitting, where the model performs well on training data but poorly on unseen data. To address this, we employed **early stopping**, a regularization technique that monitors the model's performance on a validation set and stops training when no further improvement is observed.

In our implementation, the **Dice coefficient on the validation set** was used as the metric to monitor. After each epoch, the validation Dice was computed. If the validation Dice did not improve over **10 consecutive epochs** (patience = 10), training was automatically terminated. This approach ensures that the model retains the best generalization performance without unnecessarily continuing training and risking overfitting.

A checkpoint of the model was saved every time the validation Dice improved. Therefore, even if early stopping was triggered, the saved model would correspond

to the best-performing epoch. This mechanism not only reduces training time but also guarantees that the best version of the model (in terms of validation Dice) is preserved for later evaluation and deployment.

In practice, early stopping was highly beneficial in this project, especially given the tendency of segmentation models to overfit to small details in the training data if trained for too long.

4.4.5 Post-processing Techniques

In semantic segmentation tasks—especially in road extraction, where target objects are typically thin, elongated, and sometimes discontinuous—the predicted masks produced by neural networks can often suffer from two major issues: disconnected segments and small, noisy regions. These artifacts can reduce the practical usability of predictions, especially in downstream applications like navigation or cartographic mapping. To address these problems, a lightweight post-processing step was applied to refine the raw binary predictions from the model.

The first operation applied is **morphological dilation**, a classical image processing technique used to "grow" or "expand" the white (foreground) regions in a binary image. In this context, dilation helps to bridge small gaps and reconnect fragmented road segments that may have been separated due to prediction errors. The operation uses a square kernel (typically 3×3), and the dilation is applied for a configurable number of iterations (e.g., one or two passes). This ensures that even thin road segments, which might otherwise be broken into disjointed pieces, become more connected and continuous.

Following dilation, we employ a **small object removal** procedure to eliminate isolated noise in the predicted masks. These false positives typically manifest as small white blobs in non-road regions, and they can be automatically removed using the `remove_small_objects()` function from the `skimage.morphology` module. This function scans the binary mask, identifies all connected components, and retains only those whose area exceeds a predefined threshold (e.g., 100 pixels). As a result, spurious detections that do not represent meaningful road structures are discarded.

The post-processing function operates directly on the binary output of the segmentation model and returns a cleaned-up mask that is better suited for evaluation and visualization. This two-step process—morphological dilation followed by small object removal—is computationally efficient, easy to implement, and significantly improves the visual quality and spatial coherence of the predicted masks without

requiring any retraining of the model.

4.4.6 Evaluation Procedure

The evaluation of the trained model was performed using the **held-out validation and test sets**. To ensure consistency, we applied the same preprocessing pipeline to validation/test images as during training (except for data augmentation, which was disabled).

For each input image, the model produced a **probability map** with values between 0 and 1, representing the likelihood of each pixel belonging to the road class. This probability map was then **applied a threshold value** and **a postprocessing step** to obtain a binary mask. Pixels with predicted probabilities above the threshold value were classified as "road," and others as "background."

The binarized predictions were compared to the ground truth masks using these metrics:

- **Dice coefficient:** measures overlap between predicted and ground truth masks.
- **Intersection over Union (IoU):** quantifies the ratio of intersection over union of predicted and ground truth regions.
- **Confusion matrix-based metrics:** Precision, Recall, F1-Score

These metrics were computed **per-image** and then averaged across the entire validation and test sets. In addition, we recorded the **validation loss** (combination of BCE + Dice Loss) for monitoring convergence.

Throughout the evaluation, quantitative metrics were complemented by **qualitative inspection** of predicted masks overlaid on original satellite images to assess visual accuracy, completeness of road extraction, and correctness of thin road structures.

4.4.7 Other Configurations

In addition to the optimizer, learning rate, loss function, and early stopping settings discussed previously, several other configurations were applied to the training pipeline to ensure robust model development.

The model was trained using:

- **Batch size** = 4: chosen to balance memory usage and convergence stability.
- **Number of epochs** = 50: set as an upper bound; early stopping often terminated training earlier depending on validation performance.

- **Weight initialization:** model initialized with pretrained weights from **ImageNet** (via ResNet-34 backbone) for encoder layers; decoder layers initialized randomly.

During training, data augmentation was applied **on-the-fly via a custom Dataset class**, as described in the Data Preprocessing section. No augmentation was applied at validation and test time.

The final model checkpoint (corresponding to the best validation Dice) was saved in **PyTorch `.pth` format** and used for both evaluation and deployment in downstream applications.

CHAPTER V. EXPERIMENTAL RESULTS

This chapter presents the evaluation results of our models on the road segmentation task, using both quantitative metrics and qualitative visualizations. The goal is to assess the model's ability to detect and delineate road structures from satellite imagery, both numerically and visually.

5.1 Quantitative Results

5.1.1 Evaluation Metrics Recap

As discussed in Chapter 3, two key metrics were used to evaluate the segmentation quality: the **Dice coefficient** and the **Intersection over Union (IoU)**. These metrics quantify the degree of overlap between the predicted binary mask and the ground truth annotation. The **Dice coefficient** was chosen as the primary criterion for early stopping and model selection due to its sensitivity to small structures, which is crucial for road extraction. Meanwhile, the **BCE + Dice Loss** was used during training to jointly optimize both pixel-level and region-level performance.

For post-training evaluation on the test set, we extended the metrics to include **Confusion Matrix-based metrics**, such as:

- **Precision:** the proportion of predicted road pixels that are actually roads.
- **Recall:** the proportion of actual road pixels correctly predicted.
- **F1-Score:** the harmonic mean of precision and recall.

These additional metrics are crucial in road segmentation, where both false positives (misidentifying background as road) and false negatives (missing actual road segments) are detrimental to real-world applications.

5.1.2 Performance of D-LinkNet

After training the D-LinkNet34 architecture for 50 epochs, the model achieved the following peak performance on the validation set:

Metrics	Best Dice	Best IoU	Final Val Loss	Final Train Loss	Epoch Reached
Value	0.7450	0.6147	0.3423	0.3146	50

Table 3 Results from DLinkNet

Performance steadily improved across epochs, reflecting both convergence and effective generalization. The model demonstrated a strong ability to detect both wide and narrow road segments.

Then, we evaluated the best D-LinkNet model checkpoint with and without post-processing techniques on the test set. The performance was assessed using different kernel sizes (for morphological dilation) and threshold values (for binarizing the predicted probability map).

Threshold	Dice	IoU	Precision	Recall	F1-Score
0.2	0.7497	0.6214	0.7489	0.7884	0.7682
0.3	0.7500	0.6218	0.7665	0.7727	0.7696
0.4	0.7496	0.6214	0.7787	0.7608	0.7696
0.5	0.7488	0.6204	0.7894	0.7497	0.7690

Table 4 Without Post-Processing (Threshold Sweep)

Observation: The performance remains relatively stable across thresholds between 0.2 and 0.5, with Dice around 0.749 – 0.750. The optimal threshold without post-processing is arguably 0.3 – 0.4 based on a balanced F1-score.

Kernel Size	Dice	IoU	Precision	Recall	F1-Score
1	0.7488	0.6204	0.7489	0.7884	0.7682
2	0.7439	0.6136	0.7406	0.7894	0.7642
3	0.7311	0.5977	0.6925	0.8187	0.7503
4	0.7082	0.5687	0.6447	0.8369	0.7283
5	0.6855	0.5413	0.6023	0.8514	0.7055

Table 5 With Post-Processing (Kernel Size Sweep, Threshold = 0.5)

Observation: The **best overall performance is obtained with kernel size = 1**, which achieves the highest Dice, IoU, and F1-score. Increasing kernel size further reduces precision due to more false positives, despite slight gains in recall.

Threshold	Dice	IoU	Precision	Recall	F1-Score
0.2	0.7497	0.6213	0.7491	0.7884	0.7682
0.3	0.7500	0.6218	0.7666	0.7727	0.7696
0.4	0.7496	0.6214	0.7788	0.7607	0.7697
0.5	0.7488	0.6204	0.7895	0.7496	0.7690
0.6	0.7475	0.6187	0.8004	0.7374	0.7676
0.7	0.7452	0.6158	0.8135	0.7216	0.7648
0.8	0.7405	0.6099	0.8314	0.6970	0.7583

Table 6 Threshold Sweep (Post-Processing Kernel Size = 1)

Observation: Again, threshold values between 0.3 – 0.5 deliver balanced results. Threshold = 0.3 slightly edges the others with the highest Dice and F1-score.

5.1.3 Post-Processing Configuration Summary

Based on the above experiments, we conclude that:

- **Kernel size = 1** performs best. Larger kernel sizes lead to increased false positives.
- **Threshold = 0.3** offers the highest Dice and F1-score with post-processing.
- Thus, all further qualitative analysis and deployment use **kernel_size = 1**, **threshold = 0.3**, and **dilate_iter = 1** as the default post-processing configuration.

5.1.4 Comparison with Other Models

In this section, we present a comparative analysis between the primary model used in this study, **D-LinkNet**, and a modern pretrained transformer-based model, **SegFormer-B1**, which was fine-tuned on the same dataset and evaluated using identical metrics and training configurations.

The results across 50 training epochs indicate that D-LinkNet consistently achieves higher segmentation performance compared to SegFormer-B1 in terms of Dice score and Intersection over Union (IoU).

Specifically, **D-LinkNet reached its best validation Dice score of 0.7450** at epoch 49, while **SegFormer-B1 achieved a best Dice score of 0.7075** at epoch 48. The following table summarizes the peak performance of both models:

Models	Best Dice	Best IoU	Best Val Loss
D-LinkNet	0.7450	0.6147	0.3381
SegFormer-B1	0.7075	0.5681	0.3963

Table 7 Best results from both models

While SegFormer-B1 showed **steady improvement and competitive early-stage performance**, its final accuracy plateaued earlier than D-LinkNet. This may be attributed to the model's pretrained nature on general-purpose semantic datasets (e.g., ADE20K), which differ significantly from road extraction tasks in aerial imagery. Additionally, the lightweight MLP decoder in SegFormer, although efficient, might not reconstruct fine spatial details (e.g., narrow roads) as effectively as the transposed convolution-based decoder in D-LinkNet.

It is also worth noting that D-LinkNet was trained **from scratch (except encoder)** and still managed to outperform SegFormer, highlighting the importance of the alignment between task and architecture. D-LinkNet's skip connections and dilated convolution blocks (DBlocks) seem to be particularly effective in preserving high-frequency spatial structures critical to road segmentation.

On the other hand, SegFormer offered **advantages in training speed (faster per epoch)** and **lower GPU memory usage**, making it a viable choice in resource-constrained environments or as part of an ensemble.

In summary, while **D-LinkNet remains the superior model for this task in terms of segmentation quality**, the experiment with SegFormer-B1 demonstrates the potential of transformer-based models in dense prediction and confirms the extensibility of our pipeline to support state-of-the-art pretrained architectures.

5.1.5 Training Progress

The model's training and validation curves are visualized below, offering insight into convergence behavior and the generalization gap:

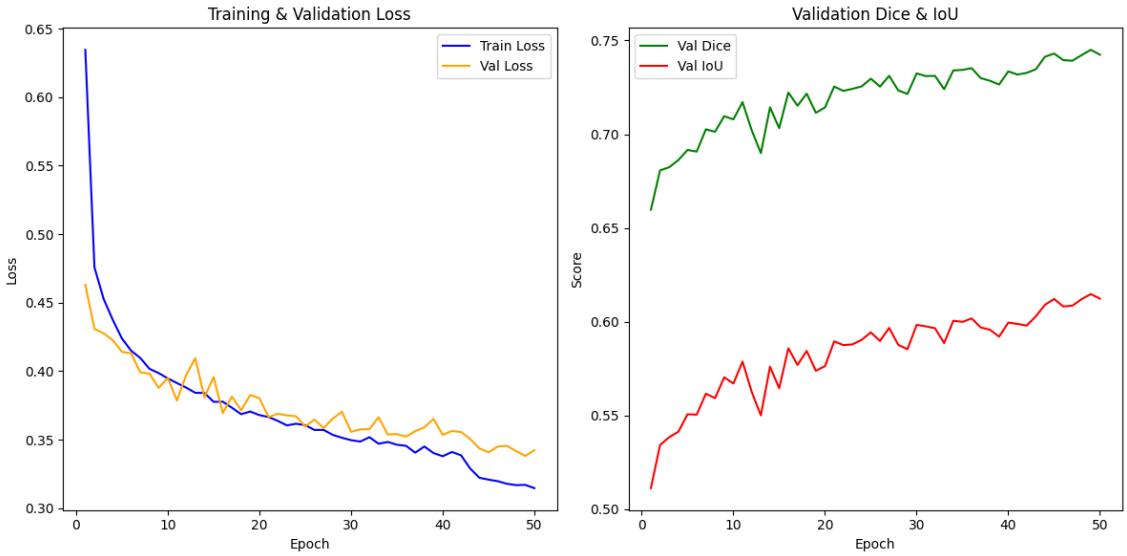

Figure 21 Training & Validation Loss, Validation Dice & IoU from D-LinkNet over epochs

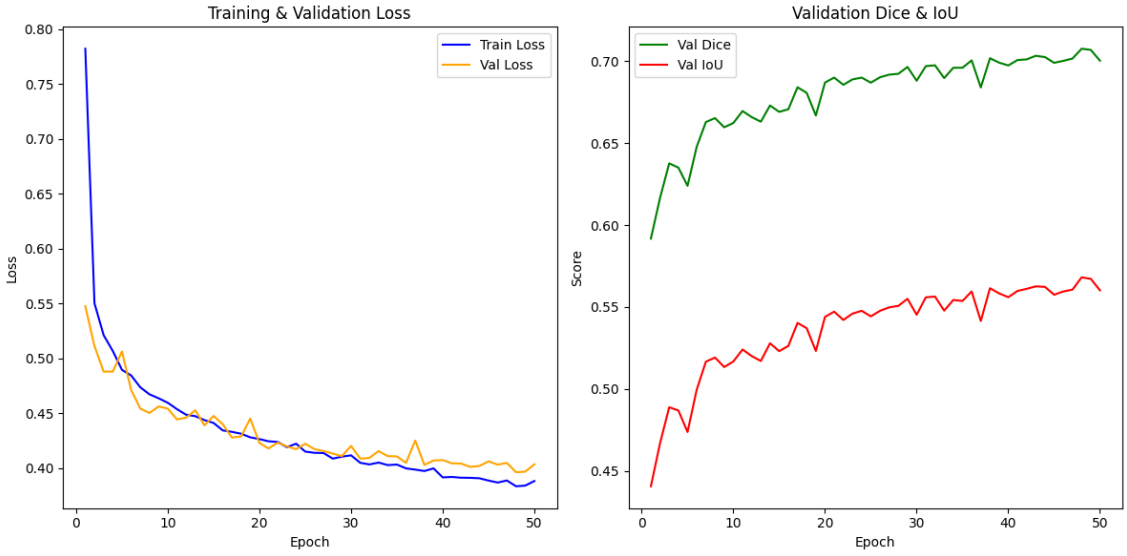

Figure 22 Training & Validation Loss, Validation Dice & IoU from SegFormer-B1 over epochs

5.2 Qualitative Results

5.2.1 Visualization Samples

We present qualitative examples to showcase the model's prediction capability on unseen validation data. Each sample includes: Input RGB image, Ground truth binary mask, Predicted mask by the model, Overlay of prediction on original image.

Here are some samples from our D-LinkNet model:

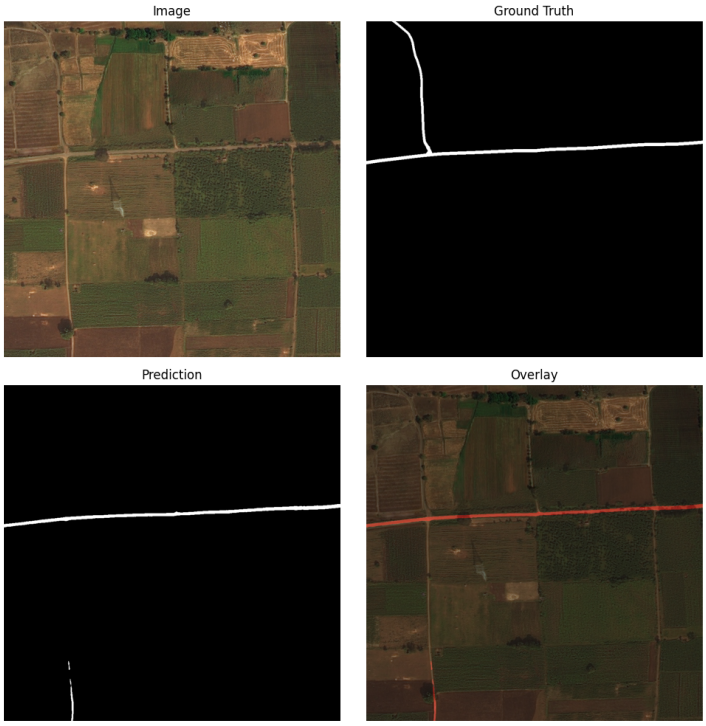

Figure 23 Sample 1 from DLinkNet

Figure 24 Sample 2 from DLinkNet

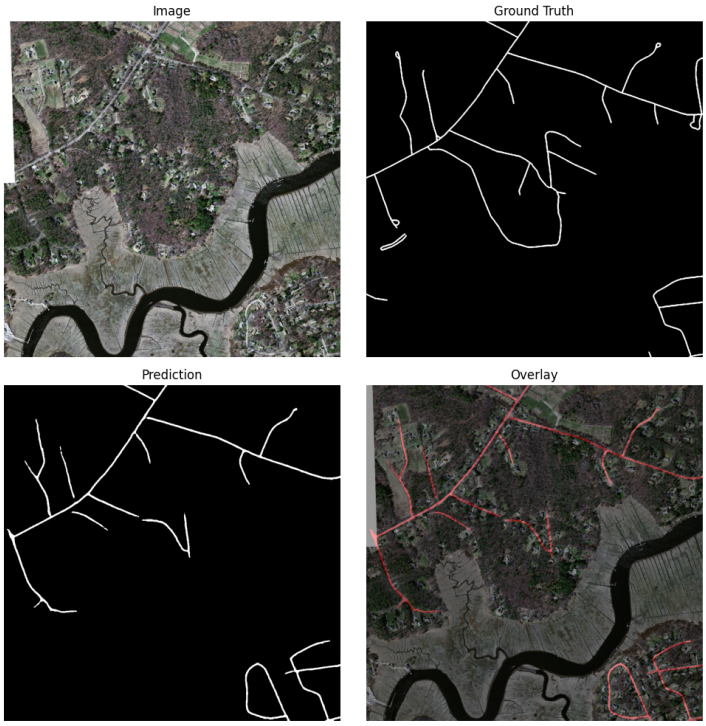

Figure 25 Sample 3 from DLinkNet

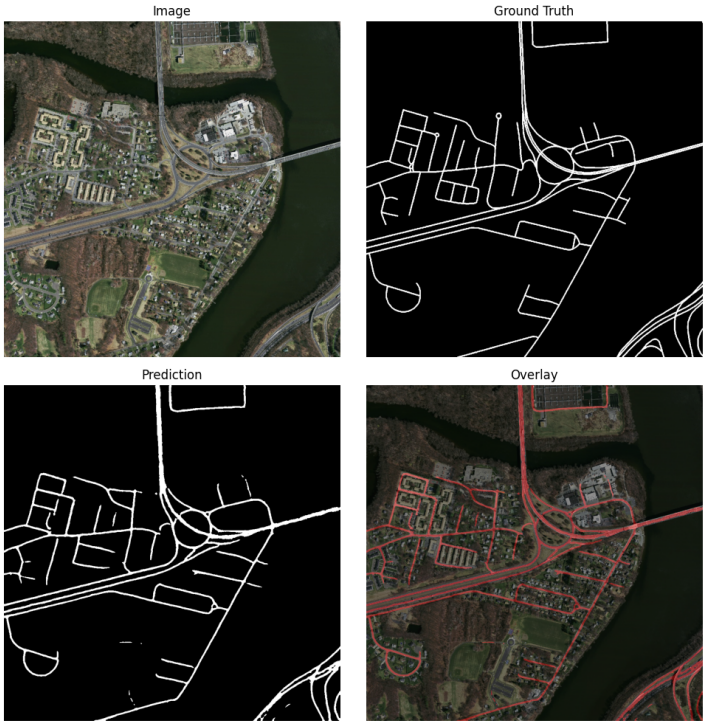

Figure 26 Sample 4 from DLinkNet

Here are some samples from our SegFormer model:

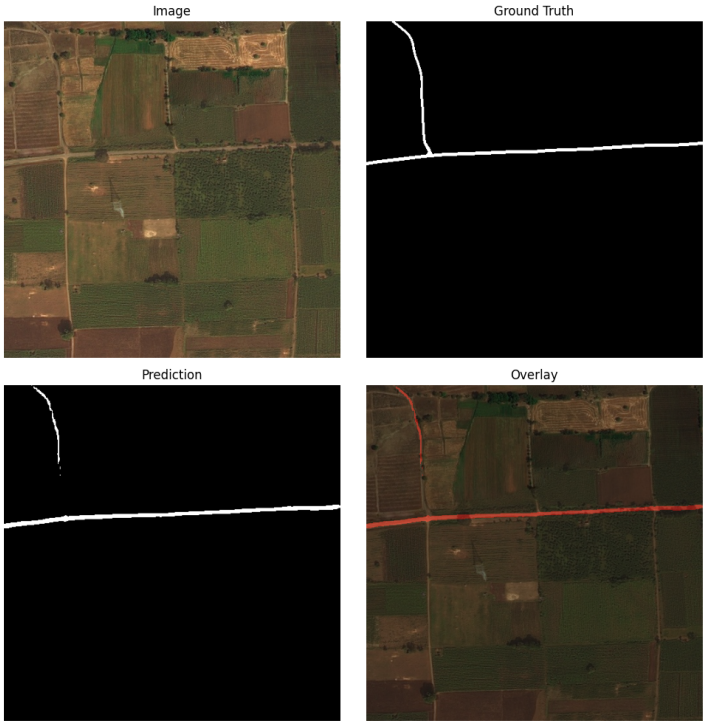

Figure 27 Sample 1 from SegFormer-B1

Figure 28 Sample 2 from SegFormer-B1

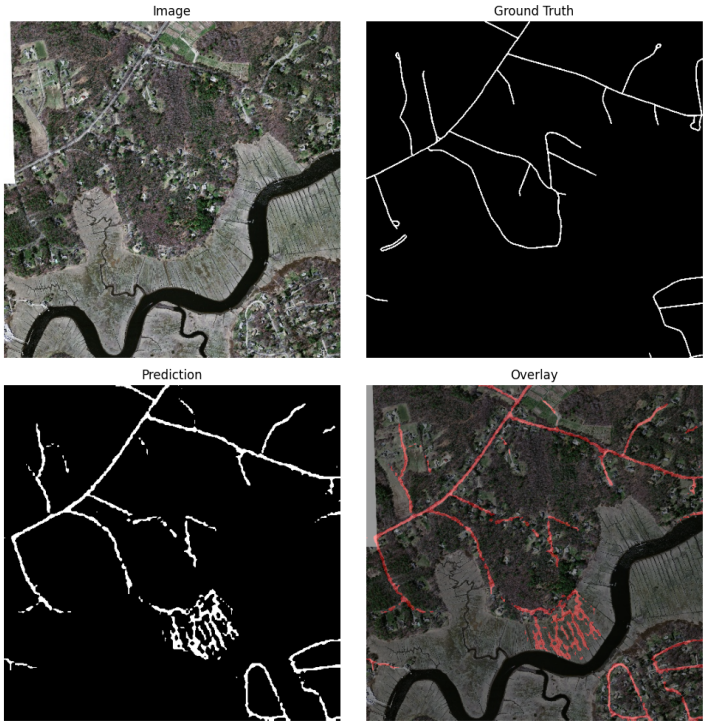

Figure 29 Sample 3 from SegFormer-B1

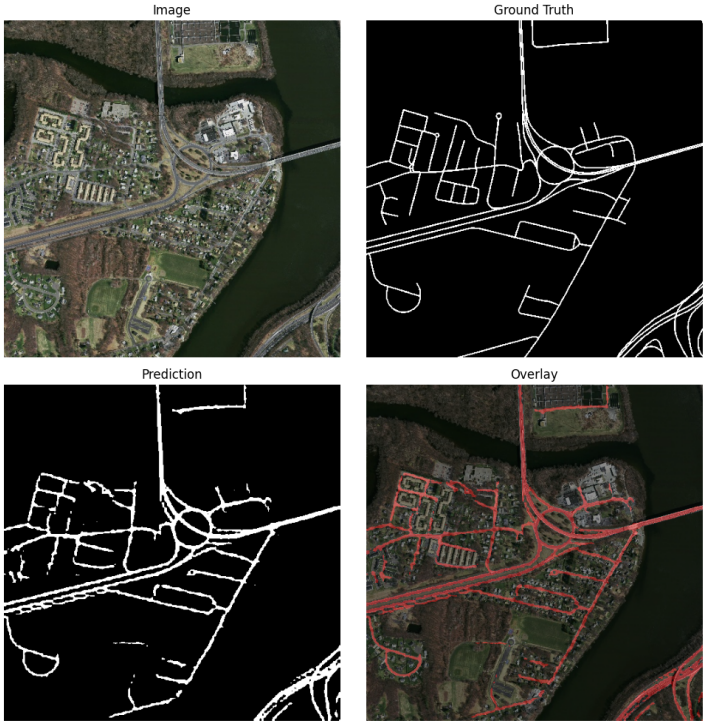

Figure 30 Sample 4 from SegFormer-B1

5.2.2 Analysis

1. D-LinkNet

The visual outputs from the D-LinkNet model demonstrate a high level of seg-

mentation quality, particularly in terms of **boundary precision** and **structural continuity** of roads. In most test images, the model successfully delineates road networks — both straight and curved — and aligns closely with the ground truth.

In the **first two samples**, the predictions are quite good. The roads are **smooth**, **well-connected**, and **aligned pixel-perfect** with the ground truth masks. This is particularly evident in rural environments where road structures are simpler and more distinguishable from the background. The overlay images confirm that the predicted masks (in red) match the white-labeled ground truth almost completely, even at intersections.

The **third sample**, which appears to be from a suburban or forested area, showcases the model's ability to handle **complex junctions and curved roads**. Even in areas where vegetation partially occludes roads, the model maintains good spatial coherence. Minor deviations can be observed at thinner road branches, but the overall topology is preserved — which is critical in applications like navigation or urban planning.

The **fourth sample**, from an urban setting with dense infrastructure, reveals D-LinkNet's strength in detecting roads amidst a cluttered scene. The predicted mask accurately outlines **roundabouts**, **parking lots**, and **interconnected paths**, indicating that the encoder-decoder structure and skip connections allow the model to retain high-frequency spatial information.

A key takeaway from these results is the model's capability to **balance fine-grained details and global connectivity**. Thanks to the **dilated convolutions** in the center block, D-LinkNet can perceive long-range context, enabling robust detection of road segments that span across the image, even under challenging lighting or terrain variations.

2. SegFormer-B1

The visual results of the SegFormer-B1 model illustrate its **strengths in capturing global context**, but also reveal certain limitations when dealing with fine-grained road structures.

In the first two test samples — representing rural and semi-rural environments — SegFormer shows **good alignment with the main road segments**. The predictions follow the primary road paths with reasonable accuracy, especially for wider or well-defined roads. However, compared to D-LinkNet, the segmentation mask tends to be **less precise at edges**, with slightly **thicker or blurred boundaries**.

In the overlay images, while the overall alignment is preserved, there are visible misalignments in narrow sections or at road endings.

In contrast, the third sample, featuring a **more complex suburban/forested scene**, challenges the model's performance. Although SegFormer correctly detects some major roads, it struggles with **small branches and occluded paths**. The prediction map reveals a noticeable amount of **false positives**, with extra detections in vegetation-rich areas or regions with texture patterns resembling roads. This suggests that while SegFormer's MiT encoder captures global information, it may lack the **fine spatial sensitivity** required for subtle road delineation — particularly when thin structures are embedded within noisy backgrounds.

The fourth sample, taken from a **dense urban setting**, presents mixed results. The model successfully segments several major roads and highways, demonstrating its ability to **generalize to large-scale structures**. Nonetheless, finer details such as narrow alleyways, parking lots, or side streets are often **incompletely detected or fragmented**. The overlay view confirms that while the main layout of the road network is captured, the overall completeness is lower than that achieved by D-LinkNet.

Overall, SegFormer-B1 performs reasonably well on road extraction tasks, particularly in cases where road features are **visually salient and continuous**. Its transformer-based encoder allows it to model long-range dependencies effectively, leading to good predictions in less cluttered scenes. However, its **lightweight MLP decoder** and lack of skip connections seem to reduce its ability to preserve high-resolution spatial detail, which is crucial for applications requiring precise road maps or topological completeness.

5.2.3 Failure Cases

This section summarizes typical failure patterns observed in both D-LinkNet and SegFormer-B1 models. Although both architectures generally perform well, several challenging cases highlight their respective limitations.

A. D-LinkNet

1. **Disconnected road segments:** In some areas, particularly those with poor contrast or visual ambiguity, the model fails to detect thin or partially occluded roads. For example, in **the second and third examples**, several minor road branches are **partially missing** or disconnected, breaking the structural continuity of the network.

2. In **the first sample**, the model mistakenly segments certain man-made rectangular structures (e.g., buildings, parking lots, or pathways) as roads due to their visual similarity in tone and shape to roads. This leads to **false positives** in the predicted mask.

3. **Edge degradation and border noise:** In the **last sample**, some road predictions near the image borders are either incomplete or slightly misaligned. This can be attributed to a lack of full context due to convolutional padding or tiling issues during inference.

These failure modes are consistent with common challenges in semantic segmentation of high-resolution aerial imagery, where **complex backgrounds**, **narrow class boundaries**, and **label noise** can affect model reliability. Although the D-LinkNet architecture with skip connections helps retain spatial detail, these cases suggest that further improvements are possible, such as:

- Incorporating attention mechanisms to focus on small road structures.
- Leveraging multi-scale feature fusion to improve detection of thin roads.
- Applying post-inference graph-based refinement to enforce road continuity.

Nevertheless, such cases are relatively infrequent and do not significantly degrade the overall segmentation quality, as evidenced by both qualitative and quantitative metrics.

B. SegFormer

Despite achieving competitive performance, SegFormer-B1 exhibits several distinct failure patterns that highlight the architectural limitations of transformer-based segmentation in high-resolution aerial imagery.

1. **Over-segmentation in complex backgrounds:** In the third test image, SegFormer-B1 mistakenly predicts dense patches of non-road areas (such as vegetation or terrain textures) as roads. This often happens when background patterns contain linear or elongated features that resemble roads. The lack of high-resolution local feature extraction in the decoder contributes to such false positives.

2. **Incomplete or broken segmentation in urban layouts:** While the model captures main road structures effectively, it often fails to detect narrow alleyways, parking lot connectors, or roundabouts in dense urban environments. As seen in the fourth sample, the predicted masks are fragmented or missing in areas with intricate infrastructure. This reflects the limitations of the lightweight MLP decoder, which lacks spatial refinement mechanisms like skip connections or multi-scale feature fusion.

3. **Coarse predictions at small-scale details:** Although SegFormer captures global structure well, it struggles to maintain fidelity at small-scale road details, especially in rural areas with thin roads or occlusions. Compared to D-LinkNet, which benefits from high-resolution skip pathways, SegFormer predictions are often slightly misaligned or disconnected at intersections and road endpoints.

4. **Blurred boundaries and width inconsistency:** The predicted masks frequently show smoothed or thickened road segments, particularly in regions where road width varies. This indicates a lack of adaptive detail recovery in the decoder and suggests that the model may generalize road patterns coarsely, especially when constrained by low-resolution intermediate outputs.

These failure cases are aligned with typical shortcomings observed in transformer-based architectures when applied to pixel-level segmentation. The MiT encoder in SegFormer is highly effective at capturing global context, but the absence of spatially aware decoding limits the model's ability to recover fine boundaries. Future improvements may include:

1. Integrating **hybrid decoders** that combine MLPs with convolutional refinement blocks.

2. Introducing **feature fusion from earlier encoder stages** to improve resolution sensitivity.

3. Applying **post-processing techniques**, such as conditional random fields or connectivity-based filtering, to enhance structural consistency in output masks.

Despite these issues, SegFormer-B1 still demonstrates strong performance in large-scale road extraction, especially in less cluttered or rural scenes. Its architectural efficiency and generalization ability make it a promising candidate for integration into multi-model or ensemble-based road segmentation pipelines.

5.2.4 Additional Discussion

1. Similar Performance with and without Postprocessing (Kernel Size = 1)

   An interesting observation arises when comparing the performance of the model **with and without postprocessing**, particularly when the **dilation kernel size is set to 1**. Quantitatively, the results are **remarkably similar** across all key metrics — Dice, IoU, precision, recall, and F1-score. For example, at a threshold of 0.5, the model with postprocessing using kernel size = 1 got a Dice score of **0.7488**, which is almost identical to the **0.7488** achieved without postprocessing. Similar patterns are observed across other thresholds.

   This similarity can be explained by the **minimal impact of dilation** when the kernel size is set to 1. Morphological dilation with a 1×1 kernel essentially performs no expansion of the mask, meaning the postprocessing step acts almost as a **pass-through** operation. Likewise, the small connected component filtering (`remove_small_objects`) typically has little effect if the predictions are already clean and continuous, as is the case with a well-trained model like D-LinkNet.

   Therefore, while the postprocessing pipeline is conceptually important and proves beneficial for larger kernel sizes (e.g., ≥ 3), it becomes **redundant when the kernel size is 1**, especially under high-quality prediction conditions. This reinforces that **the network alone can achieve near-optimal segmentation** if appropriately trained and regularized.

2. Application-Oriented Threshold Selection: When to prioritize Precision or Recall

   Choosing an appropriate threshold for binarizing the model's probability map is **highly dependent on the application domain**. As demonstrated earlier, lower thresholds (e.g., 0.3) tend to lead to **higher recall**, while higher thresholds (e.g., 0.7 or 0.8) increase **precision** at the cost of missing true road pixels.

   (a) When to Prioritize Precision

   In domains where **false positives (FP)** can lead to critical failures, **precision must be prioritized**. For instance:

   - **Navigation Systems:** Predicting a sidewalk or a river as a road (FP) can mislead GPS systems and pose safety risks.
   - **Autonomous Vehicles:** False roads could be misinterpreted by self-driving algorithms, leading to **accidents**.

In these cases, it's preferable to **miss some road segments (FN)** rather than include incorrect ones. A **higher threshold (e.g., 0.6 – 0.8)** should be used, ensuring that only high-confidence predictions are retained. Additionally, **postprocessing techniques** (e.g., small object removal) can help filter out noise, further improving reliability.

(b) When to Prioritize Recall

In contrast, some applications value **completeness** of the road network, even if some predictions are incorrect:

- **Emergency Response:** In disaster zones, missing a road could prevent rescue teams from accessing a critical area.
- **Urban Planning:** When mapping roads for infrastructure development, it is essential to capture **all possible paths** (even small or obscured ones).

Here, **false positives are tolerable** as long as **true roads are not missed**. Lower thresholds (e.g., 0.2–0.3) are more suitable, ensuring high recall. Though this may increase noise, postprocessing can still be applied to clean the predictions without sacrificing coverage.

The samples below are applied the postprocessing step with **super low threshold values**, proving the ability to capture obscured ones which are not marked as official roads in the ground truth, despite the fact that some non-road pixels are oversegmented as roads.

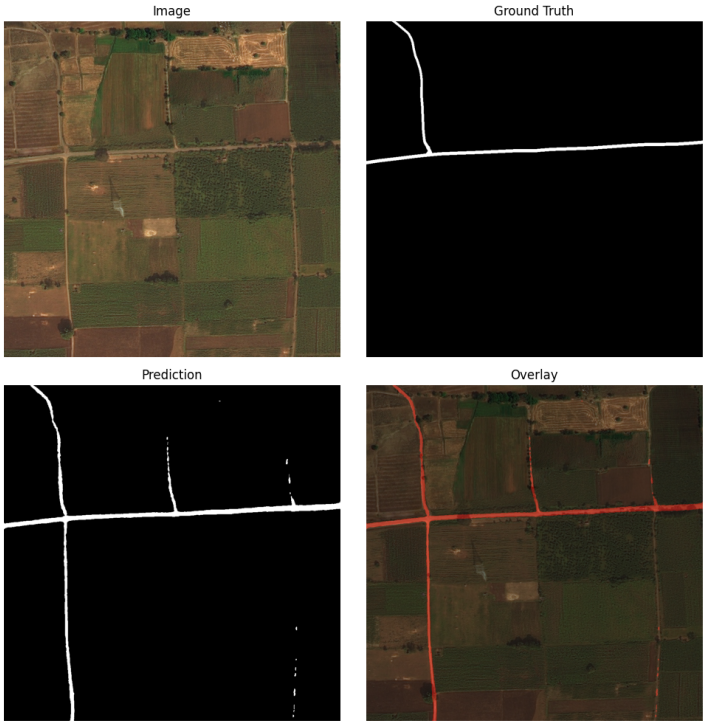

Figure 31 Threshold = 0.001

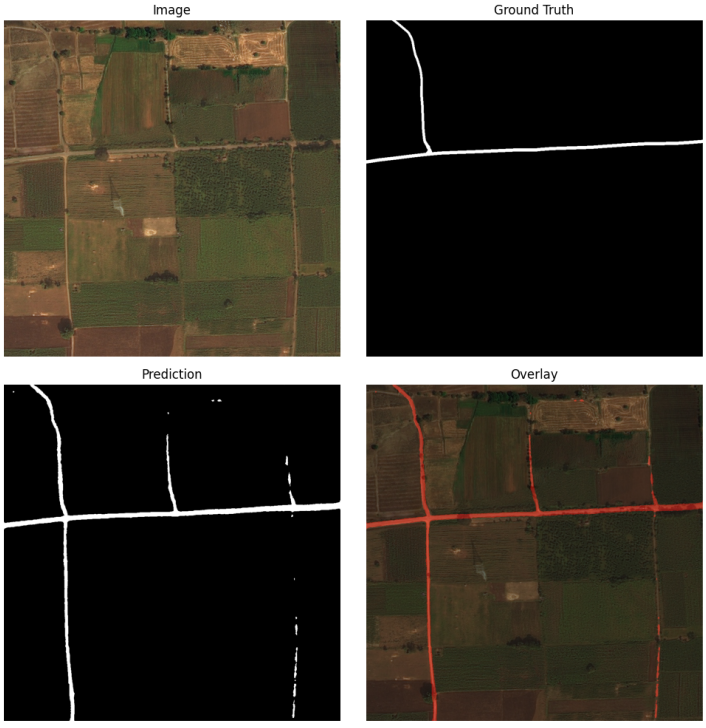

Figure 32 Threshold = 0.0005

This understanding enables practitioners to tune the threshold and postprocessing dynamically based on real-world needs—striking the right trade-off between completeness and accuracy.

CHAPTER VI. CONCLUSION

6.1 Summary of Findings

In this project, we successfully implemented and trained the **D-LinkNet** model for road segmentation on high-resolution satellite imagery. Leveraging its encoder-decoder architecture with skip connections and dilated convolutions, D-LinkNet achieved strong performance on the validation set, with a **best Dice coefficient of 0.7450** and **IoU score of 0.6147**.

Our data pipeline proved effective in unifying multiple datasets and performing consistent preprocessing and augmentation. The integration of normalization, advanced loss functions (BCE + Dice), and learning rate scheduling helped ensure training stability and convergence. The model demonstrated robustness across both rural and urban scenes, producing sharp and coherent road masks.

In addition to D-LinkNet, we also fine-tuned a **pretrained SegFormer-B1** model to evaluate the effectiveness of transformer-based architectures for the same task. SegFormer-B1, despite using a lightweight MLP decoder and lacking explicit spatial skip connections, showed competitive performance, reaching a best **Dice score of 0.7075** and **IoU of 0.5681**. Its MiT encoder successfully captured global context, particularly in scenes with large, continuous road structures.

However, SegFormer-B1 exhibited limitations in segmenting fine road details and narrow branches, especially in complex urban or forested environments. Nonetheless, its efficient architecture, faster training per epoch, and lower memory footprint make it a viable candidate for deployment in resource-constrained scenarios or as part of a multi-model ensemble.

Overall, both models demonstrated the effectiveness of deep learning-based approaches in road extraction, with D-LinkNet excelling in boundary precision and SegFormer offering a strong balance between accuracy and efficiency.

6.2 Limitations

Despite its strong performance, the model shows limitations in certain scenarios:

- Roads occluded by shadows or vegetation are sometimes missed or fragmented.
- Narrow alleyways and complex intersections may be under-segmented.
- Predictions in visually ambiguous regions occasionally include false positives.

- While D-LinkNet struggles primarily due to the limited receptive field of CNNs, SegFormer-B1's lightweight MLP decoder lacks the spatial refinement necessary to capture fine-grained details, leading to blurred or incomplete segmentations.
- SegFormer-B1 also exhibits coarser outputs due to the lower resolution of decoder features, which may result in width inconsistency or edge misalignment without post-processing.

These limitations highlight the challenges of semantic segmentation in high-resolution aerial imagery and the trade-offs between architectural complexity and prediction precision.

6.3 Future Work

Several directions can be explored to further enhance model accuracy and generalization:

- **Attention mechanisms** (e.g., SE blocks, CBAM) may improve feature selection in D-LinkNet and help focus on road-relevant structures under occlusion.
- **Multi-scale feature fusion techniques**, such as feature pyramid networks (FPN) or spatial pyramid pooling, could enhance robustness to scale and road width variations.
- **Post-processing methods**, including Conditional Random Fields (CRFs), connected component analysis, or morphological operations, may help refine boundaries and enforce connectivity.
- Further **fine-tuning of transformer-based architectures** like SegFormer (e.g., using deeper variants such as B2 or B3) may improve long-range reasoning while addressing decoder resolution limitations.
- Combining both models in a **hybrid or ensemble framework** could leverage the boundary precision of CNNs with the contextual awareness of transformers.
- Exploring **self-supervised pretraining or domain adaptation** may also improve performance when generalizing to new geographic regions or satellite providers.

In conclusion, this study demonstrates the feasibility and effectiveness of using both **CNN-based (D-LinkNet)** and **transformer-based (SegFormer-B1)** models for road segmentation. While D-LinkNet provides high spatial accuracy, SegFormer offers a scalable and efficient alternative. Together, they lay a strong foundation for future research and real-world deployment in remote sensing, urban planning, and autonomous navigation.

REFERENCE

[1] V. Mnih, "Machine learning for aerial image labeling," Ph.D. dissertation, University of Toronto, 2013.

[2] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[3] [Online]. Available: https://github.com/twigsWHy/dlinknet

[4] M. W. Lichen Zhou, Chuang Zhang, "D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018, pp. 182-186*, 2018.

[5] F. H. Ilya Loshchilov, "Decoupled weight decay regularization," *7th International Conference on Learning Representations, New Orleans*, 6-9 May 2019.

APPENDIX. MEMBER DISTRIBUTION

I. DATA PROCESSING

Phùng Tiến Thành

- Choosing the appropriate datasets for our topics.
- Perform images & masks preprocessing.
- Apply augmentation methods on our dataset.

II. PROGRAMMING TASK

D-LinkNet built from scratch - Phùng Tiến Thành & Đinh Bảo Hưng
Finetune SegFormer - Nguyễn Phan Thắng

III. SIDE TASK

Write report - Phùng Tiến Thành & Đinh Bảo Hưng
Slides design - Nguyễn Phan Thắng